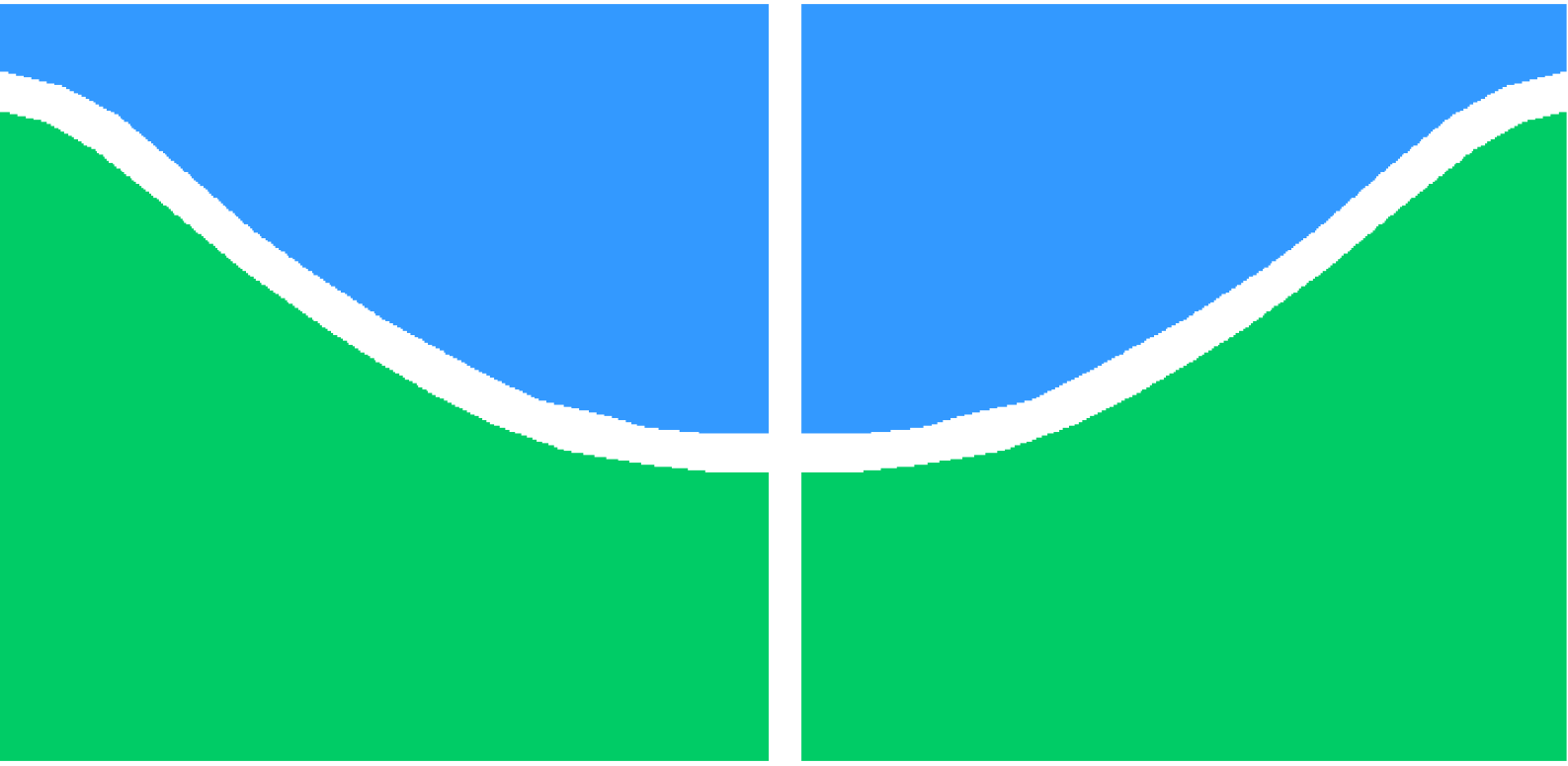

Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Estudo sobre infraestruturas seguras de votação utilizando BlockChain

Autor: Matheus Miranda Lacerda
Orientador: Prof. Dr. Tiago Alves da Fonseca

Brasília, DF
2019

Matheus Miranda Lacerda

Estudo sobre infraestruturas seguras de votação utilizando BlockChain

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Tiago Alves da Fonseca

Brasília, DF

2019

Matheus Miranda Lacerda
Estudo sobre infraestruturas seguras de votação utilizando BlockChain/ Matheus Miranda Lacerda. – Brasília, DF, 2019-
56 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Tiago Alves da Fonseca

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2019.

1. BlockChain. 2. E-Voting. I. Prof. Dr. Tiago Alves da Fonseca. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Estudo sobre infraestruturas seguras de votação utilizando BlockChain

CDU 02:141:005.6

Matheus Miranda Lacerda

Estudo sobre infraestruturas seguras de votação utilizando BlockChain

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 11 de Julho de 2019 – Data da aprovação do trabalho:

Prof. Dr. Tiago Alves da Fonseca
Orientador

Dr. José Antônio Carrijo Barbosa
Convidado 1

Eng. Tomás Malheiros Borges
Convidado 2

Brasília, DF
2019

Agradecimentos

Gostaria de agradecer primeiramente à Deus e à minha família, por todo o suporte em todos os momentos e por me apoiar na minha jornada de estudo ao longo desses anos, em especial à minha amorosa mãe Mariluse Miranda, ao meu trabalhador pai Osmar Lacerda e à minha irmã Anna Carolina Miranda.

Agradeço também ao meu orientador Prof. Dr. Tiago Alves da Fonseca, por todo o direcionamento e auxílio no desenvolvimento deste trabalho.

Deixo também a minha gratidão à todos os meus companheiros do laboratório LAPPIS, por todo o imenso aprendizado, companheirismo e diversão.

Por último, agradeço imensamente à minha professora Dra. Carla Rocha de Aguiar por me aconselhar e guiar no meu desenvolvimento profissional e humano, como uma grande amiga e mestre.

Resumo

As *Blockchains* são tecnologias implementadas como uma base de dados distribuída. Estas tecnologias se tornaram conhecidas com a popularização da criptomoeda *Bitcoin*, uma vez que esta criptomoeda funciona a partir da utilização da *Blockchain* como mecanismo principal para armazenamento e gerenciamento de transações.

O motivo desta tecnologia ser utilizada como base de dados pela maioria das criptomoedas é que devido à sua implementação, as *Blockchains* possuem a característica de serem imutáveis, e com elas é possível garantir a integridade dos dados armazenados. Estas propriedades fizeram com que estas fossem exploradas para diversos contextos além do uso monetário comum. Foram, então, desenvolvidas diversas criptomoedas análogas à *Bitcoin*, para aplicações com necessidades relacionadas à auditabilidade e segurança dos dados armazenados.

Uma das áreas onde o uso das *Blockchains* vem sendo explorado é a de construção de soluções seguras para realização de processos de votação, onde garantir aspectos de auditabilidade e integridade são extremamente importantes.

Neste trabalho estuda-se a utilização da tecnologia *Blockchain* para a implementação de uma infraestrutura segura de votações, que permita que os usuários possam auditar seus votos a fim de garantir que estes são armazenados e não podem ser alterados. É feita uma análise a respeito do uso da *Blockchain* neste contexto, justificando os aspectos viáveis de serem implementados, e consequentemente elencando os aspectos positivos e fragilidades do uso da tecnologia para solução do problema em questão.

Palavras-chaves: Blockchain, E-voting, Ethereum.

Abstract

Blockchains are technologies implemented as a distributed database. These technologies have become well known with the popularization of the crypto-currency Bitcoin, as this crypto-currency works by using Blockchain as the primary mechanism for transaction storage and management.

The reason for this technology to be used as a database by most crypto-coins is its characteristic of being immutable, and with them it is possible to guarantee the integrity of the stored data. These properties have made them exploited for different contexts beyond the common monetary use. A number of crypto-currencies analogous to Bitcoin have been developed for applications with needs related to the auditability and security of stored data.

One of the areas where the use of Blockchains is being explored is the construction of secure solutions for conducting voting processes, where ensuring auditing and integrity aspects are extremely important.

This work studies the use of Blockchain technology to implement a secure voting infrastructure, allowing users to audit their votes to ensure that they have been stored and have not changed. An analysis is made of the use of Blockchain in this context, justifying the feasible aspects of being implemented, and consequently listing the positive aspects and weaknesses of the use of technology to solve the problem in question.

Key-words: Blockchain, E-voting, Ethereum.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

O desenvolvimento acelerado da tecnologia, o respectivo aumento do volume de informações coletado e o processado, aliado ao amadurecimento da sociedade contemporânea, tornaram as falhas dos modelos de votação atuais mais visíveis e amplamente divulgadas. No Brasil, as votações nacionais para cargos eleitorais são realizados utilizando urnas eletrônicas que utilizam códigos fechados. Que não são diretamente acessíveis para auditabilidade por toda a comunidade.

Paralelamente, a discussão cresceu ao longo dos anos sobre a utilização de processos eletrônicos para automatizar e melhorar os processos de votação, e assim buscar mitigar seus principais desafios. Vários países utilizaram diversas abordagens de votação eletrônica, e com as experiências realizadas perceberam a complexidade de tal desafio e o quanto era necessária a criação de novos paradigmas para resolução deste problema (KERSTING, 2004).

Com a popularização das Criptomoedas, tornaram-se mais populares também as *Blockchains*. Muitas pessoas enxergaram o potencial desta tecnologia para construção de soluções descentralizadas que ajudem a dissolver problemas de segurança e dependabilidade, uma vez que visam possuir o mínimo de partes externas interferindo nos seus processos. Estas pessoas então passaram a utilizar essa tecnologia para construção de diversos tipos de aplicações descentralizadas, entre elas aplicações para realização de processos de votação (SWAN, 2015).

As *Blockchains* são redes distribuídas formadas por cadeias de nós, que trabalham colaborativamente para validação de dados através de protocolos pré-definidos implementados sobre a arquitetura P2P(Peer-to-Peer). A implementação e utilização da arquitetura peer-to-peer para criação de aplicações distribuída é anterior à criação das *BlockChains*. Porém, a inovação que a tecnologia *Blockchain* possibilitou foi permitir a implementação de protocolos distribuídos onde os nós não precisam confiar plenamente um nos outros na execução destas aplicações (RAVAL, 2016).

Através da execução contratos distribuídos programáveis, as *Blockchains* garantem o correto funcionamento da rede e colaboração dos nós. Estes contratos são característicos por serem armazenados dentro da própria *Blockchain*, em vez de serem armazenadas dentro de um servidor (RAVAL, 2016).

Neste trabalho é feita uma introdução aos principais aspectos técnicos das redes *Blockchain*, e uma análise à respeito de processos de votação eletrônica. De acordo com os aspectos fundamentados, foi realizada a discussão sobre a viabilidade da criação de uma infraestrutura segura para votações utilizando a tecnologia *Blockchain* como principal

ferramenta.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo principal deste trabalho é planejar e desenvolver uma infraestrutura para votações utilizando *BlockChain*. Esta estrutura será focada em resolver requisitos de auditabilidade em relação aos votos, garantindo que estes possam ser armazenados e auditados de forma segura.

1.1.2 Objetivos específicos

1. Será feito um estudo técnico das tecnologias *Blockchain*, de modo a verificar e atestar quais dos requisitos de segurança definidos são viáveis e podem ser efetivamente implementados a partir da utilização destas.
2. Ao fim do trabalho, pretende-se analisar os principais desafios e problemas que ainda precisam ser solucionados para a implementação da solução proposta e da solução ideal.

1.2 Organização do trabalho

A organização deste trabalho se deu da seguinte forma:

- **Fundamentação Teórica:** Esta seção aborda os principais aspectos à respeito de soluções de votação eletrônica (*E-voting*), bem como seus desafios e benefícios;
- **Metodologia:** Esta seção explica a forma como esse trabalho será realizado. Além disso, nesta seção é definido o escopo da solução proposta;
- **Resultados:** Esta seção apresenta os resultados parciais do trabalho, mostrando a validação de alguns aspectos da solução. Aqui também são definidos os trabalhos futuros, aspectos da solução que não foram abordados nesta etapa do trabalho.

2 Fundamentação Teórica

2.1 Votação Eletrônica(*E-voting*)

O termo *E-voting*, caracteriza qualquer processo ou sistema para votação que utiliza processos ou dispositivos eletrônicos, seja completamente ou parcialmente. Desta forma, encaixam-se dentro do contexto de *E-voting* desde os sistemas via Internet até os dispositivos de urnas eletrônicas utilizadas em eleições em vários países do mundo (KERSTING, 2004).

Apesar das diversas abordagens utilizadas atualmente, a maioria dos processos de votação eletrônica utiliza algum procedimento associado ao uso da Internet, mesmo que em níveis baixos. Os contextos de votação podem ser classificados quanto à sua dependência da Internet, uma vez que, de acordo com o nível dessa dependência, é possível identificar certas características e necessidades. No geral, quanto mais o sistema possui partes de sua arquitetura dependentes da Internet, menos este mesmo sistema depende da interferência e regulação de diversas partes.

2.1.1 Impacto da utilização de sistemas de *E-voting*

Os estudiosos que defendem a utilização de sistemas de votação eletrônicos em detrimento dos processos não eletrônicos elencam alguns argumentos principais (KERSTING, 2004):

- Democracia barata: A democracia poderia ser mais barata, uma vez que realização de processos de votação em diversos países, em geral envolvem grandes custos relacionados à aspectos de segurança, organização, e etc;
- Contagem mais segura e eficiente: No geral a utilização de procedimentos eletrônicos em contextos de votação fornecem uma contagem rápida e eficiente de votos. Partindo do pressuposto que os sistemas eletrônicos utilizados possuem confiabilidade e integridade, estes provavelmente farão uma contagem e apuração dos votos de forma mais confiável e rápida do que quando o mesmo processo é realizado manualmente;
- Aumento na participação eleitoral: O aumento direto na participação eleitoral pode ser uma consequência. A partir do momento que existem diversos meios mais amigáveis para realização de votações, é possível aumentar o interesse de certos nichos da população. Além disso, a descentralização e facilitação de votações por meio da utilização de dispositivos eletrônicos pode ser de grande impacto em zonas de conflito ou regiões de difícil acessso;

- Votações mais justas: Com o processo de votação sendo implementado eletronicamente, é mais fácil implementar métodos de votação que são mais justos em relação ao contexto em questão. Como exemplo, a utilização de votos parciais, votos de preferências, e etc;

2.1.1.1 Impacto no preço da democracia

A economia direta de dinheiro em contextos mais políticos, como eleições, pode se tornar um problema no contexto de *E-voting*. Como existe uma grande quantidade de dinheiro envolvida em todo o processo com contratação de serviços, produção de dispositivos, etc, muitas partes não estariam a favor, por exemplo, da utilização de aplicações de *software* livre, ou da eliminação da utilização de dispositivos proprietários (KERRNEL, 2016). Descentralizar e democratizar processos de votação implica diretamente na descentralização do poder sobre o processo de votação, o que afeta o lucro relacionado.

Um outro aspecto que trouxe receio aos usuários durante o desenvolvimento de soluções de *e-voting* é que algumas das mais famosas aplicações implicavam em custos adicionais para os cidadães - custos de tempo, complexidade maior, gastos com compra de dispositivos diferentes - o que causa pouca aderência por parte deles aos novos paradigmas de votação. E, muitas vezes, a própria experiência ao testar métodos diferentes de votações não foi positiva para os eleitores (KERSTING, 2004).

Baratear o processo de votação pode ser um ganho que acontece como consequência direta de processo de *E-voting*. Porém, como elencado por Kerrnel(2016), a preocupação direta com custos deveria ser muito menos relevante do que a preocupação direta com a qualidade do processo ou serviço sendo utilizado: como o foco de uma eleição deve ser que esta seja o mais segura e confiável possível, não faz sentido violar aspectos de um processo em questão para economia de dinheiro.

2.1.1.2 Consequências na contagem dos votos

Em relação a eleições que utilizam cédulas em papel, sabe-se que sempre haverá um grande risco de leitura errada durante o processo de votação em si, ou contagem errada durante o processo de apuração dos votos. Além disso sempre há o fator de aumento de tempo — e consequentemente de recursos — durante os processos de votação e apuração devido a questões de limitação humana (PAN; ANSARI, 2011).

Em função desse risco, a contagem dos votos está sempre a cargo de uma terceira parte nos processos de votação atuais. O que implica que o nível de segurança e confiabilidade da contagem só é tão elevado o quanto se confia em quem está realizando a contagem dos votos.

Assim como na solução proposta por Ahmed e Xiaowen(2013) que depende de um

centro de registros para armazenagem, contagem e auditabilidade dos votos, os sistemas de votação demandarão uma genuína preocupação com a segurança física da infraestrutura onde os votos estão armazenados. Além disso, quanto mais tempo os dados estão armazenados sobre a custódia de terceiros, maior é o risco de vazamentos de informações e quebra de privacidade, o que torna difícil saber por quanto tempo os votos precisam estar armazenados para auditabilidade antes do armazenamento se tornar crítico à segurança do sistema.

2.1.1.3 Impacto na participação eleitoral

A Estônia foi o primeiro país do mundo a permitir que seus eleitores realizassem seus votos via *Internet* para eleições parlamentares em 2007. A partir dos resultados coletados na eleição, foi feita uma análise do fluxo de votação entre 2003 e 2007 para estimar se a utilização do processo de votação via *Internet* realmente influenciou no aumento da participação eleitoral (BOCHSLER, 2010). A conclusão foi que houve um aumento na participação eleitoral total do país, porém esse crescimento não esteve necessariamente relacionado à disponibilidade de votação via *Internet*. Entretanto, como mostrado em Fig. 1, houve um percentual de efeito de substituição, com o voto eletrônico substituindo o processo tradicional de votação.

No Brasil, também foi realizado um processo de análise de aumento da participação popular com a introdução de um mecanismo de votação via *Internet*. Este processo foi realizado durante o evento anual de Participação para Votação de Despesas realizado do Rio Grande do Sul. O resultado foi que houve um aumento total de 8,2 por cento na participação de votantes (SPADA JONATHAN MELLON; SJOBERG, 2015). Não houve presença significativa de efeitos de substituição que indicassem a migração de paradigmas de votação por parte dos eleitores. O que indica que o meio de votação eletrônico atraiu para o processo uma parcela da população que não possuía um grande contato com processos políticos ou que não se sentiam motivados para ingressarem na participação (SPADA JONATHAN MELLON; SJOBERG, 2015).

2.1.1.4 As votações se tornam mais justas?

Existem diversos tipos de processos de votações válidos. Explorar métodos de votações diferentes, em diferentes contextos e condições sociais, é um exercício importante e necessário. Na grande maioria dos países, os processos de votação de grande importância, como eleições presidenciais, sempre utilizam o sistema de votação conhecido como *Plurality Voting*, em que os eleitores escolhem apenas um candidato para apoiar. O problema disto é que os processos de votação possuem casos de falhas que podem levar a resultados eleitorais injustos em relação à democracia, que não representam a vontade da maioria (ARROW, 1951).

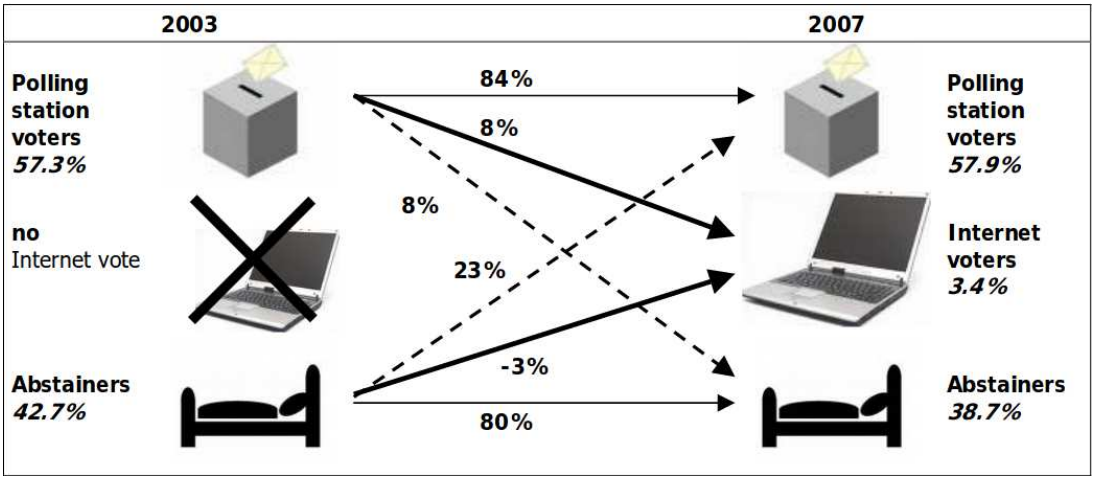

Figura 1 – Fluxo de votos na Estônia de 2003 a 2007. (BOCHSLER, 2010)

Analisando os processos em um contexto de eleições para cargos políticos, o sistema de votação *Plurality Voting* sofre do efeito de *Spoiler*. Efeito que acontece quando um ou mais candidatos que possuem pautas muito parecidas recebem parcelas dos votos de um mesmo público, uma vez que eleitores com opiniões parecidas dividirão os votos entre os candidatos, o que dará uma vantagem para candidatos que se oponham à ambos (Paul Cuff, Sanjeev Kulkarni, Mark Wang and John Sturm, 2015). Além disso, eleitores podem preferir votar em candidatos que não são sua opção preferida, apenas para evitar que um candidato indesejado ganhe. O que pode fazer com que os votos que deveriam ir para os candidatos preferidos sejam destinados ao candidato que pode evitar que os candidatos menos desejados sejam eleitos.

Uma alternativa válida seria utilizando o processo de *Preferencial Voting*, onde os eleitores podem prover uma lista dos candidatos que apoiam, ordenados de acordo com a preferência. Ainda no contexto de eleições de políticos, este método pode ser muito mais representativo e eficaz que a abordagem anterior. Porém, supondo um contexto onde houvessem três candidatos, chamados de C1, C2 e C3, Caso as listas de preferências {C1, C2, C3}, {C2, C3, C1} e {C3, C1, C2} recebessem o mesmo número de votos, seria impossível determinar o candidato preferido pela maioria.

Arrow define que para que uma eleição seja considerada plenamente democrática, os seguintes três critérios devem ser seguidos (ARROW, 1951):

1. Caso todos os eleitores prefiram uma alternativa, então essa é a melhor alternativa;
2. Os eleitores podem escolher livremente as alternativas;
3. Ninguém pode escolher sozinho a preferência de todo o grupo.

Arrow(1951) também demonstra que nenhum processo de votação que possua mais

de dois candidatos pode ser completamente democrático. Ou seja, este pode chegar à um caso em que o resultado não represente efetivamente a vontade da maioria: um dos três critérios precisará ser violado em algum momento do processo eleitoral. Um exemplo da quebra de uma dessas regras para resolução de impasses é a utilização de eleições de segundo turno, realizadas apenas entre os dois candidatos mais votados. Neste caso, a regra número dois é quebrada, e os eleitores não podem escolher livremente entre todas as opções.

A justificativa da utilização de um processo de E-voting está justamente na facilidade de flexibilização, pois é significativamente menos custosa a mudança de paradigma de votação em um sistema eletrônico baseado na *Internet* do que em um sistema baseado em processos mais físicos e dependentes de mais partes envolvidas. Para sistemas baseados em processos físicos, poderia ser necessário maior gasto com adequação da população, compra de novos dispositivos, mudança nos protocolos de segurança, mudanças na legislação, e *etc.*

Apesar de não ser possível elaborar um processo de votação plenamente democrático, é importante que haja a utilização de diversos métodos de votação, de acordo com o contexto e a necessidade, com o objetivo de minimizar as falhas, e fazer com que os resultados finais representem da melhor maneira possível a opinião da maioria dos votantes.

2.1.2 Aspectos de segurança de sistemas de *E-voting*

Dentro de um contexto de *E-voting* totalmente seguro e confiável, os principais conceitos de segurança são aplicáveis da seguinte forma (HASSAN, 2013):

- Confidencialidade: O dispositivo em questão deve garantir que apenas o cidadão pode visualizar o próprio voto.
- Anonimidade: Não deve ser possível mapear a identidade do cidadão através de seu voto.
- Integridade: Deve haver garantia de que os votos foram devidamente registrados. E que uma vez registrados, estes não serão alterados.
- Autenticidade: A autentificação do votante deve ser garantida no dispositivo através de um esquema de segurança que ao mesmo tempo que garanta que o cidadão está apto a votar, não permita o rastreamento deste através de sua própria "assinatura" utilizada para votar.
- Verificabilidade: Deve haver uma maneira de verificar os votos quando necessário, e identificar quando algum deles foi violado.

- Auditabilidade: Deve existir uma forma de auditar os votos sem que para isso seja revelada a identidade dos votantes.

2.1.3 *E-voting* via Internet

Desde 2006, muitos pesquisadores estudam soluções de Ciência da Computação relacionada à criação de estruturas, modelos e possíveis soluções para *E-voting*. Porém, quanto mais se desenvolveram pesquisas à respeito, mais temerosos ficaram os pesquisadores face aos numerosos e crescentes desafios de segurança da Internet. Muito desse temor se deve às soluções anteriores de implementações de *E-voting*, que não eram maduras o suficiente para superação de tais desafios (HASSAN, 2013).

Na maioria dos sistemas de *E-voting* baseados na Internet, não há um mecanismo que possa garantir que o voto do usuário foi devidamente armazenado e computado e que, ao mesmo tempo, não exponha explicitamente a identidade ou voto dos usuários.

Entre os sistemas de *E-voting* que possuem a propriedade de auditabilidade – o que implica que existe um mecanismo que permite que os votos sejam auditados – uma das soluções mais populares e estudadas foi o sistema chamado Helios (HASSAN, 2013).

Um dos principais elementos da implementação do Helios, que pode ser um empecilho em relação a alguns aspectos de segurança, é a necessidade de utilização de um navegador por parte do cliente. Muitas das aplicações que são baseadas na utilização da *Internet* provavelmente possuirão esta característica. Porém, os *browsers* são sistemas que não foram construídos com o foco em serem utilizados como um *Trusted Software*, porém, foram construídos com o foco em possuir usabilidade máxima por parte dos usuários, buscando para tanto também ser o mais extensível e flexível possível. O que torna mais difícil identificar previamente e sanar vulnerabilidades, por ser um sistema complexo dependente de muitas outras partes. Um exemplo desta lógica é o *plugin* Flash Player, que é utilizado na maioria dos navegadores modernos. Esse *plugin* foi desenvolvido para executar praticamente qualquer formato de vídeo e possui diversas vulnerabilidades (HASSAN, 2013), o que o torna um artefato de operação incompatível com navegação segura.

Um dos princípios para o desenvolvimento de aplicações deste tipo deve ser uma lista diminuta de *Trusted Softwares*, ou seja, poucas partes em que o usuário deve confiar plenamente para utilização das aplicações. Quanto mais dependências houver do lado do cliente, menos este terá que confiar que esta mesma ferramenta funciona de forma segura.

Em contradição ao aspecto anterior, retirar do alcance do usuário *Trusted Parties* que eventualmente poderiam facilitar a interação deste com o processo de votação poderia prejudicar criticamente a usabilidade deste mesmo processo. É claro perceber, por exemplo, que, para uma grande gama de usuários de diversas faixas etárias e condições sociais, seria mais simples a utilização de um *Browser* para realizar a votação em

vez de um programa *client* específico desenvolvido para ser instalado no computador do usuário. O processo de utilizar um *client* mais restrito na máquina do usuário poderia potencialmente deixar o sistema mais seguro, porém este também ficaria consequentemente, mais complexo e menos acessível a pessoas com menos conhecimento tecnológico, ou que não possuam recursos para aquisição de dispositivos necessários para utilização de tais ferramentas.

2.2 Tecnologia *BlockChain*

2.2.1 Histórico da tecnologia *BlockChain*

Apesar de fontes indicarem que a ideia da tecnologia possa ter sido desenvolvida alguns anos antes, esta começou a ser amplamente difundida e utilizada a partir de 2008 com a divulgação do *paper* de criação da criptomoeda BitCoin (NAKAMOTO, 2008). Neste *paper*, a blockchain foi proposta como a principal arquitetura a ser implementada para o funcionamento correto da criptomoeda, afim de solucionar problemas como o gasto duplo de moedas.

A motivação do autor(NAKAMOTO, 2008) surge a partir de sua análise da utilização de dinheiro em aplicações eletrônicas. Este discorre em seu texto que um problema inerente deste tipo de aplicação está na alta dependabilidade de terceiros para regulação de operações e gerenciamento monetário. Além disso, como tais sistemas são baseados em confiança, o nível de informações fornecidas por cada parte para alcançar a confiança mútua aumenta. Foi proposto então um sistema que, em vez de depender diretamente da confiança entre as partes, fosse baseado na utilização de prova criptográfica e que pudesse oferecer mecanismos de garantia para as duas partes, com transações que fossem dificilmente reversíveis.

2.2.2 O que é a tecnologia *BlockChain*?

A *BlockChain* é uma base de dados distribuída de transações armazenadas em blocos de informação. A cadeia de blocos distribuída na rede *BlockChain* é chamada de *ledger*, e é mantida sincronizada por todos os nós participantes, ao longo de uma rede *peer-to-peer*. A coleção de blocos é crescente, com a adição de novos blocos sendo feitas ao final da cadeia.(BRAGA; SANTOS, 2017). Além disso, o armazenamento dos dados é rastreado ao longo do tempo e todos os blocos são guardados desde sua criação.

O comportamento da Blockchain e a maneira como os nós interagem entre o si são determinados por protocolos de consenso. Protocolos de consenso são o fator que garantem a integridade da *ledger*, e é onde são definidos aspectos como a maneira como os nós da

rede devem se comportar e colaborar na criação de blocos, confiabilidade e políticas de transações.

Cada bloco de informação adicionado à *Blockchain* possui uma *hash* do bloco de informações anterior. Desta forma, os nós devem trabalhar para atingir uma concordância sobre quais informações, e, principalmente, a ordem em que estas informações foram inseridas na cadeia. E o protocolo de consenso garantirá que os nós concordem a respeito da ordem como as informações estão armazenadas na *ledger* distribuída (CACHIN, 2017).

Como levantado por Rifa e Budi(2017), mesmo com os diversos tipos de *BlockChain* e protocolos de consenso, no geral as implementações de *BlockChain* apresentam algumas características comuns, sendo elas:

- Arquiteturalmente, a *BlockChain* é descentralizada e distribuída ao longo da rede *peer-to-peer*, o que diminui ou elimina a necessidade de terceiros no gerenciamento da cadeia;
- Sua distribuição acontece ao longo dos nós em tempo real;
- O processo de autenticidade em uma rede *BlockChain* se dá pela utilização de técnicas criptográficas;
- Curiosamente, as redes *BlockChain* não provêem um mecanismo que impede diretamente a alteração de dados. Porém os protocolos utilizados na *BlockChain* são definidos de forma a fazer com que o processo de alteração de dados seja extremamente difícil, ou impossível, de ser realizado em tempo hábil. No caso da Bitcoin, por exemplo, é preciso atingir o consenso por parte de mais de 50 por cento dos nós da rede.
- O rastreamento de informações dentro da rede é baseado no tempo;
- Os protocolos e mecanismos de funcionamento da *BlockChain* podem ser programados, o que caracteriza uma boa flexibilidade e extensibilidade.
- Como os dados estão armazenados ao longo de todos os nós da rede, geralmente as aplicações *BlockChain* possuirão uma alta disponibilidade;
- A verificabilidade e integridade dos dados também são pontos fortes, uma vez que estes são objetivos primários da *BlockChain* e consequências diretas de sua implementação.

2.2.3 Características e funcionamento da *BlockChain*

Alexandre, Fernando e Robson(2017) dividem as *BlockChains* em dois grupos principais, sendo o primeiro grupo formado pelas *Blockchains* abertas ou não permissionadas.

No primeiro grupo, encaixam-se criptomoedas como o Bitcoin e o Ethereum. Neste tipo de *Blockchain* qualquer um pode se tornar um nó e começar a escrever na *ledger* criando novos blocos, ou participar do consenso para criação. Além disso, nesse modelo de rede, os nós não precisam confiar plenamente uns nos outros, uma vez que estes são competidores diretos na criação dos blocos. O funcionamento da rede depende apenas da boa execução dos protocolos de consenso por parte dos participantes.

O segundo grupo é formado pelas *Blockchains* privadas ou permissionadas, que são características de *BlockChains* geralmente utilizadas em contextos corporativos. Neste modelo, o acesso ou visualização dos dados é controlado e permitido apenas a usuários ou nós que estejam devidamente autorizados e autenticados. Como os usuários não são totalmente anônimos, as regras de confiança são baseadas em outros aspectos como permissões, papéis, e etc. Tais aspectos definem a forma como o consenso será realizado e como a *BlockChain* funcionará(BRAGA; SANTOS, 2017).

2.2.4 Transações

Cada bloco na cadeia é formado por um conjunto de transações que foram processadas por um determinado nó, agregadas em um bloco e espalhadas ao longo da rede como um possível candidato a bloco final.

No caso das Criptomoedas, as transações no geral são formadas a partir do padrão: um *timestamp* da transação, uma *hash* que identifica a transação anterior, um valor de entrada, o valor de saída da transação, o endereço de destino do valor e uma assinatura digital criada a partir da chave privada do usuário que criou a transação(BRAGA; SANTOS, 2017).

As transações em uma rede Bitcoin possuem o formato descrito na Tabela 1.

Cada campo possui o seguinte significado:

- `versão`: Versão das transações sendo utilizadas, o que determina a versão do protocolo de consenso sendo utilizado pelos nós.
- `tx_in count`: Número de transações entradas relacionadas à transação.
- `tx_in`: Transações de entrada da transação.
- `tx_out count`: Número de transações de saída relacionadas à transação.
- `tx_out`: Transações de saída da transação.
- `lock_time`: *Timestamp* Unix da transação. O que indica o tempo mínimo de criação que um bloco deve ter para incluir tal transação como parte do bloco.

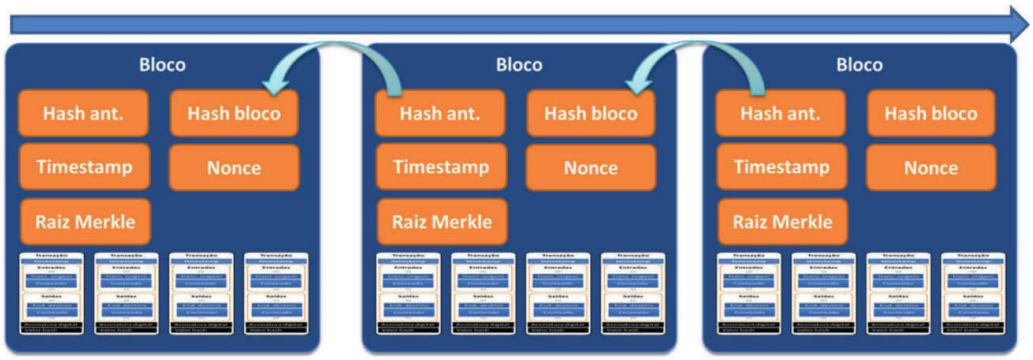

Figura 2 – Cadeia de blocos. (BRAGA; SANTOS, 2017)

Tabela 1 – Descrição dos dados uma transação (Bitcoin Developer Guide, 2014).

Nome	Bytes	Tipo de dado
`versão`	4	`uint32_t`
`tx_in count`	`variável`	`compactSize uint`
`tx_in`	`variável`	`txIn`
`tx_out`	`variável`	`compactSize uint`
`tx_out`	`variável`	`txOut`
`lock_time`	4	`uint32_t`

Apesar do formato da transação especificar o contexto da aplicação em questão, não existe nenhum dado dentro da *ledger* que identifique uma criptomoeda em específico, o que caracteriza o funcionamento de uma moeda em questão serão os seus protocolos de funcionamento.

2.2.5 Blocos

Os blocos de transações estão organizados na forma de uma cadeia, em uma estrutura de lista, onde cada bloco possui uma referência direta com o bloco vizinho imediatamente anterior. Como mostrado na Figura 2 dentro do cabeçalho de um Bloco $B[x]$ sempre haverá uma referência à *hash* do bloco anterior $B[x-1]$. O principal intuito desta referência é que caso qualquer informação seja alterada no bloco anterior a *hash* de bloco deste bloco mudará, e não estará mais em concordância com o valor de *hash* armazenado no bloco atual.

A Tabela 2 mostra uma descrição da entrada de dados que forma o cabeçalho de um bloco. Os cabeçalhos dos blocos possuem um tamanho fixo de 80 bytes.

A versão do bloco indica o conjunto de regras de consenso a serem seguidos para validação do bloco.

A *Hash* do cabeçalho do bloco anterior é uma *hash* SHA256 gerada a partir do cabeçalho do bloco anterior. O que garante que o bloco anterior não pode ser modificado sem alterar o conteúdo do bloco atual.

A *Hash* da raiz *Merkle* é gerada a partir da hash de todas as transações do bloco. O que garante que nenhuma transação pode ser modificada sem que o valor da raiz seja alterada.

O tempo é um *Timestamp* Unix que representa o momento em que o nó minerador começou a gerar a *hash* do cabeçalho. Os nós completos não aceitarão blocos com tempos de cabeçalho mais de duas horas no futuro.

O campo *nBits* representa uma versão codificada da dificuldade alvo a qual a *hash* deste bloco deve ser menor ou igual.

O valor de *Nonce* é um número arbitrário que os nós mineradores, aqueles que validam as transações e criam blocos, encontram para modificar o valor do cabeçalho, com o objetivo de produzir uma *hash* menor ou igual à dificuldade alvo. Caso todos os valores de 32 *bits* sejam testados, o tempo pode ser modificado, ou a transação base do bloco pode ser modificada para alterar o valor da raiz *merkle*.

Dentro de cada bloco, as transações são organizadas a partir de uma estrutura chamada de árvore *Merkle* (NAKAMOTO, 2008; BRAGA; SANTOS, 2017). Esse tipo de estrutura é implementado através de uma árvore binária em que os nós armazenam um valor de *hash* criado a partir dos dados da transação. O valor de cada um dos nós interiores da árvore é calculado a partir do valor do *hash* dos "nós filhos". O processo de geração da árvore é feito a partir dessa estratégia *bottom-up* e termina quando o nó raiz da árvore é criado, sendo que o valor do nó raiz será uma *hash* gerado a partir da agregação das *hashes* de todos os nós da árvore.

Tabela 2 – Descrição do cabeçalho de um bloco (Bitcoin Developer Guide, 2014)

Nome	Bytes	Tipo de dado
`versão`	4	`int32_t`
`Hash do bloco anterior`	32	`char[32]`
`Hash da raiz`	32	`char[32]`
`Tempo`	4	`uint32_t`
`nBits`	4	`uint32_t`
`nonce`	4	`uint32_t`

Na criação do Bloco, apenas o *hash* da raiz da árvore é utilizada no cálculo do *hash* do bloco. O objetivo da Árvore Merkle é garantir a verificação da integridade dos

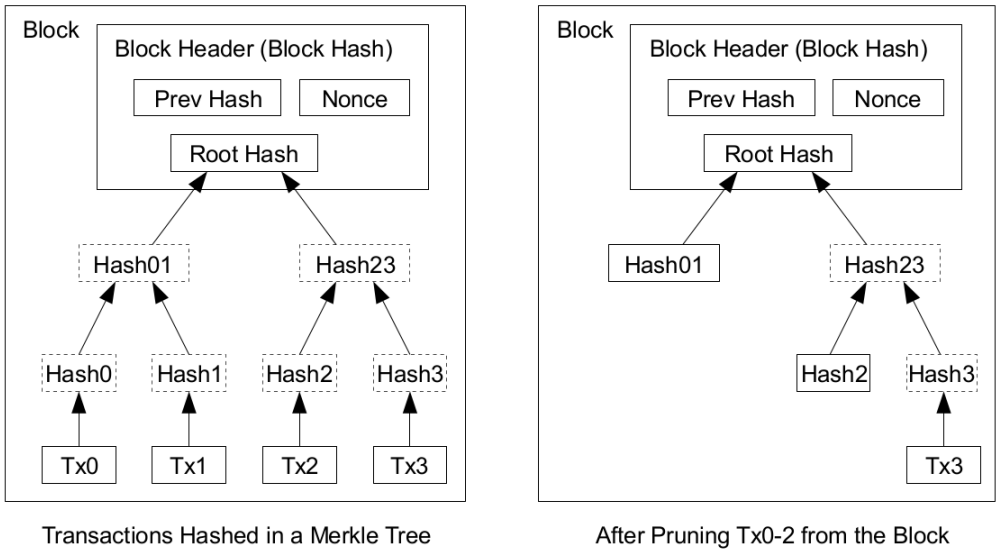

Figura 3 – Árvore Merkle. (NAKAMOTO, 2008)

dados armazenados em seus nós de forma eficiente, por isso sua utilização foi proposta por Nakamoto(2008). A Figura 3 exemplifica uma árvore criada a partir de um conjunto de transações Tx0, Tx1, Tx2 e Tx3. É possível perceber que caso o conteúdo de alguma das transações seja alterado, isso implicará na alteração consequente de todos os log(n) nós no caminho até o nó raiz, com n sendo o número de nós na árvore, o que irá então alterar o valor da *hash* no nó raiz da árvore e invalidará o bloco, que por sua vez invalidará todos os blocos da cadeia que foram criados a partir do bloco em questão e que possuem *hashes* criadas a partir da informação do bloco que foi corrompido. A agregação das *hashes* na árvore é feita em pares e quando o número de transações é impar, a hash da última transação é repetida e agregada consigo mesma. Como esta árvore é binária e "completa", caso alguma das transações seja mudada de lugar, isso também invalidará o nó raiz e o bloco a que a transação pertence.

As Árvores Merkle no contexto das criptomoedas como Bitcoin e Ethereum são utilizadas por tipos de nós chamados *thin clients*, que são nós mais simples e que não sincronizam a BlockChain inteira localmente, como fazem os *full nodes*. O propósito de tais clientes é se conectar aos *full nodes* ao longo da rede para verificar se uma transação é válida e se pertence a um determinado bloco.

O *thin client* se conecta ao *full node* e solicita a ele uma estrutura chamada *Merkle Block*, que é uma versão simplificada do bloco de transações. Este Merkle block contém um número pequeno de *hashes*, algumas informações a respeito da forma como a árvore deve ser estruturada e o cabeçalho do bloco original onde está contida a *hash* do nó raiz da árvore do bloco. Com os *hashes* e o conjunto de informações recebidas, o *client* reconstitui uma pequena parcela da árvore proporcional à $Log(n)$ do total de nós, e verifica se o *hash*

gerado para o nó raiz corresponde àquela definida no cabeçalho do bloco. Desta forma a validade de uma transação pode ser verificada de forma eficiente, também com tempo proporcional a $Log(n)$ do número de nós, além de não precisar possuir toda a cadeia de blocos armazenada localmente (NAKAMOTO, 2008; BRAGA; SANTOS, 2017).

O cabeçalho do bloco é onde estão as informações de identificação do bloco, que serão utilizadas para validação e durante a construção da cadeia. Dentro do cabeçalho do bloco se encontram:

- o *hash* de identificação do próprio bloco;
- o *hash* da raiz da árvore de transações do bloco;
- um número chamado *Nonce* que representa o valor encontrado pelo nó para provar que este realmente realizou esforço computacional considerável na mineração do bloco;
- uma referência ao *hash* do bloco anterior da cadeia;
- e um *timestamp* da criação do bloco.

O algoritmo para validação do bloco se dá da seguinte forma (Ethereum White Paper, 2014):

1. Verifica se o bloco anterior, referenciado no cabeçalho do bloco atual, existe e é um bloco válido;
2. Verifica se o *timestamp* do bloco atual é maior que o do bloco anterior. E que a diferença de tempo entre estes dois *timestamps* é de no máximo 15 minutos;
3. Verifica que a prova de trabalho contida no bloco é valida;
4. Sendo S[0] o estado no fim do último bloco da cadeia e TX a lista de transações. Para cada uma das i transações, o estado S[i+1] será definido como APPLY[S[i], TX[i]]. Ou seja, a esse estado será associado o resultado de aplicação da transação. Se qualquer uma das aplicações retornar um erro ou se o limite de *Ether* deste bloco for atingido, ocorrerá um erro;
5. S[n] será o estado final para aquele bloco, e adicionará uma transação de recompensa pelo bloco para o minerador;
6. Verifica se a raiz da árvore *Merkle* no estado final é igual ao valor definido no cabeçalho do bloco;

A Figura 4 exemplifica a construção dos estados na *ledger* durante o processo de validação de um bloco na rede *Blockchain*.

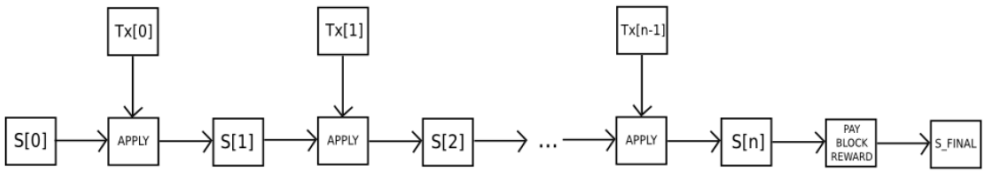

Figura 4 – Grafo de estados da *Blockchain Ethereum*. (Ethereum White Paper, 2014)

2.2.6 Mecanismos de Consenso

Dentro da rede *peer-to-peer* (P2P) das *Blockchains*, os nós interagem entre si na criação, validação de transações, criação de blocos, e etc. Porém ao mesmo tempo em que a *BlockChain* implica na ausência de terceiros para regulação do funcionamento da rede, esta implica que haja um padrão de comportamento dos nós e que estes possuam um certo nível de confiança entre si para o bom funcionamento da rede. Para tal, as *BlockChains* são implementadas com a utilização de mecanismos de consenso, baseado em protocolos que atuam como regras de funcionamento para a rede P2P.

A ideia principal dos mecanismos de consenso é construir entre os nós uma interação que seja tolerante à falhas não se baseando diretamente apenas na confiança plena entre os nós e levando em conta o fator de que é possível que uma parte da rede seja composta por nós maliciosos ou defeituosos. Ademais, tais partes maliciosas podem causar entropia no funcionamento da rede, seja de maneira proposital ou acidental.

A segurança dos protocolos de consenso é baseado na suposição de que a maioria dos operadores está mais interessada em seguir o consenso do que tentar quebrar as regras. Para tal, a maioria das criptomoedas utiliza recompensas e taxas a serem pagas aos nós para atrair mais colaboradores para a rede (BRAGA; SANTOS, 2017).

Dentro do contexto dos mecanismos de consenso, os nós que não se comportam como nós honestos – nós mal intencionados ou funcionando incorretamente – são chamados de Nós Bizantinos (BASHIR, 2017).

As *Blockchains* são altamente dependentes dos mecanismos de consenso uma vez que o funcionamento destes está relacionado à resolução, ou não resolução, dos principais problemas encontrados em sistemas distribuídos, que são a coordenação de funcionamento dos nós e a tolerância à falhas. A *Blockchain* como um sistema distribuído está diretamente preocupada com três propriedades principais que são características de todo sistema distribuído(BRAGA; SANTOS, 2017):

- Consistência: As informações mantidas entre os nós estão consistentes e condizentes entre si, de forma que todo nó possui a informação correta mais atualizada em um

determinado momento do tempo;

- Disponibilidade: O sistema está funcionando corretamente sempre que for requisitado;
- Tolerância à partição: O sistema continua operando e funcionando corretamente, mesmo que uma determinada parte da rede pare de funcionar ou comece a se comportar de maneira maliciosa;

Construir um sistema que atinja um nível satisfatório nas três propriedades é extremamente difícil. Por isso geralmente os sistemas tendem a sacrificar a maturidade em alguma das propriedades para melhorar a aplicação das outras duas propriedades. Nas *Blockchains* a replicação de todos os dados da *ledger* por todos os nós aumenta a maturidade do sistema em relação à tolerância à falhas. Além disso, em *Blockchains* não permissionadas a junção de novos nós à rede é incentivado, uma vez que qualquer um pode se tornar um nó e ter toda a cadeia de blocos sincronizada facilmente, o que também colabora para a manutenção de tal aspecto.

O uso dos mecanismos de consenso está justamente associado à garantia de consistência dos dados entre todos os participantes da rede. Porém a consistência plena entre os nós é um fator que não é trabalhado pelas redes *Blockchain*, uma vez que esta propriedade é lesada em prol da disponibilidade e da tolerância à falhas. Isso implica que, em determinados momentos haverá, momentaneamente, divergência de concordância sobre a maior cadeia válida entre os nós, até que a situação de conflito seja resolvido e os nós estejam em acordo sobre o último bloco válido, a partir do qual devem começar a concatenar novos blocos (BRAGA; SANTOS, 2017).

Imran (BASHIR, 2017), define alguns requisitos que determinam se um consenso em uma rede Blockchain é passível de ser realizado e atinge os objetivos propostos:

- Concordância: Todos os nós honestos devem chegar em um acordo comum sobre o consenso;
- Término: Todos os nós honestos eventualmente terminarão a execução do protocolo de consenso. Ou seja, todos os nós alcançarão alguma decisão em um período finito de tempo;
- Validade: O valor resultante do consenso entre os nós, deve ser igual ao valor proposto por pelo menos um dos nós;
- Tolerância à falhas: O protocolo de consenso deve funcionar corretamente mesmo na presença de nós maliciosos;
- Integridade: Cada nó pode alcançar apenas uma decisão em cada ciclo de consenso;

É preciso lembrar que o consenso não depende necessariamente da concordância de todos os nós. O mecanismo em questão determinará a validade do acordo. Como um consenso de maioria em alguns casos, ou consenso decidido por nós de acordo com seus respectivos papéis, em outros casos.

2.2.7 Blockchains públicas ou não permissionadas

Estas utilizam mecanismos de consenso baseados em de prova ou liderança em que a decisão é feita a partir de um nó escolhido pelos demais nós, ou por um processo que mostra que o nó cumpriu determinados requisitos para a criação do novo bloco de dados. A Bitcoin utiliza um mecanismo chamado Prova de Trabalho, já o Ethereum permite também a utilização de outros mecanismos como Prova de Participação e Prova de Autoridade. A ideia principal da Prova de Trabalho é que a validação de um bloco depende de um custo computacional para resolução de um determinado problema. Este problema está associado à utilização de algoritmos de geração de *hashes*. Os *hashes*, em geral, são algoritmos determinísticos para geração de números pseudo-aleatórios a partir de uma entrada de dados. Por serem determinísticos, os algoritmos de *hash* se tornam importantes para as Blockchains, uma vez estes possuem a propriedade de não inversibilidade, não sendo possível determinar o dado de entrada a partir da *hash* de saída. Assim, um algoritmo de *hash* sempre resultará na mesma *hash* sempre que o mesmo dado de entrada for passado para este, e gerará uma saída diferente caso seja mudado pelo menos um *bit* de informação do dado de entrada.

Para a geração dos *hashes*, a rede da Bitcoin utiliza um parâmetro chamado de dificuldade. Esta dificuldade é estabelecida como um valor máximo para a *hash* de validação do bloco, que consequentemente implica que a representação binária do *hash* a ser encontrado deve possuir uma determinada quantidade de zeros à esquerda.

O incentivo para que mais nós adentrem à rede, e gastem seus recursos computacionais validando transações e blocos está em um mecanismo de recompensa em que para cada bloco validado corretamente e aceito no consenso, o nó validador receberá uma quantidade X de moedas. Tanto na rede da Bitcoin quanto do Ethereum, durante o processo de mineração dos blocos de transações, a primeira transação criada no bloco é um tipo especial de transação chamada *CoinBase Transaction*, onde o nó envia para si mesmo uma recompensa correspondente a uma determinada quantidade da criptomoeda em questão.

A cada 2016 blocos a dificuldade é recalculada pela rede, de forma que o tempo esperado de trabalho acumulado para criação desta quantidade de blocos seja de aproximadamente duas semanas. A rede então recalcula a dificuldade baseado no tempo ideal, de forma a tentar garantir que este tempo ideal será atingido na mineração dos próximos 2016 blocos (Bitcoin Developer Guide, 2014). A ideia deste tempo ideal é que a resolução do problema para encontrar um bloco válido leve em torno de 10 minutos, uma vez que

este é o tempo estimado pelos criadores da Bitcoin para geração continuada da criptomoeda até que esta alcance o número de 21 milhões de Bitcoins geradas. A dificuldade da rede é então calculada segundo a seguinte fórmula:

$$Dificuldade = DificuldadeAnterior * TempoDeMineração/2016 * 10minutos$$

Onde o tempo de mineração é o tempo gasto para minerar os 2016 blocos em questão.

Esta prova resolve o problema dos conflitos a respeito de qual cadeia de blocos é verdadeira. Os nós consideram como verdadeira a cadeia que possui mais prova de trabalho acumulada. A partir dos *timestamps* definidos na criação dos blocos é possível estimar quanto recurso computacional foi gasto na construção de uma parte da cadeia.

A prova de trabalho da Bitcoin exige que o nó mostre o quanto trabalhou para a construção do bloco. A *Blockchain* fará com que cada bloco referencie o hash do bloco anterior e utilize esta na criação de seu próprio cabeçalho. Isto implica quem, para a propagação de um bloco alterado, o Nó Bizantino precisaria gerar mais prova de trabalho que a quantidade acumulada desde a criação do nó original até o atual momento no tempo. Consequentemente cada nó malicioso terá que trabalhar significativamente mais para a modificação de blocos passados e coerção de nós honestos. Quanto mais a Blockchain aumenta mais difícil é para um nó malicioso alterar toda um cadeia de nós. Em compensação, os nós que optam por validar blocos de maneira honesta precisam gastar recursos apenas na criação de novos blocos a serem adicionados ao final da ledger (Bitcoin Developer Guide, 2014).

A introdução do mecanismo de prova de trabalho por Nakamoto(2008) na implementação da *Blockchain* foi uma ótima abordagem, pois isto resolve diretamente dois problemas intrínsecos da rede. Primeiro, proporciona uma forma para que um conjunto de nós possa chegar a um consenso, de moderada eficiência e em um período finito de tempo. Segundo, cria um mecanismo de participação livre em que qualquer um pode entrar na rede e participar ativamente no consenso com igual influência sobre decisões de acordo, sem que sejam necessárias outras partes para regulação de influência e participação (BASHIR, 2017).

2.2.8 BlockChains permissionadas

As *Blockchains* permissionadas só podem ser validadas por usuários com permissões e apenas estes participam efetivamente do processo de validação dos blocos. Como apenas os participantes conhecidos e com permissão adequada possuem poder de decisão no consenso, assume-se que estes são confiáveis e não há necessidade de um sistema de provas na execução do protocolo de consenso (SEVEREIJNS, 2017).

Blockchains permissionadas surgiram com a necessidade de utilização da tecno-

logia em contextos mais controlados, onde existem um conjunto de regras restritos e os participantes não dependem da confiança uns nos outros, como em contextos empresariais (VUKOLIć, 2017).

As *Blockchains* permissionadas se dividem em dois tipos principais(SEVEREIJNS, 2017), privadas e públicas:

Blockchains permissionadas privadas:

As *Blockchains* permissionadas privadas não são distribuídas, são centralizadas e o acesso à visualização e criação de transações é controlado (SEVEREIJNS, 2017).

Estas são bem empregadas em contextos de grupos pequenos que lidam com dados sensíveis. Porém, são menos seguras em relação à integridade da rede, uma vez que um atacante ao conseguir credenciais de acesso à rede possuirá grande poder de influência.

Suas principais vantagens são em relação à privacidade e ao custo (SEVEREIJNS, 2017). Caso haja um cuidado efetivo com às permissões dos participantes, os dados possuirão um nível de privacidade muito alto.

Aliado a isso, o conjunto de regras pode ser alterado pelos nós donos da *Blockchain* com o objetivo de otimizar os recursos gastos na validação dos blocos, por exemplo, protocolos de validação em que um nó tenta validar um bloco apenas depois que este já foi validado por outros nós com mais permissões na cadeia. Além disso, os blocos precisam ser validados por um conjunto menor de nós, o que exige custos operacionais e infraestrutura significativamente menores.

Em contextos de *Blockchains* privadas permissionadas, quanto menos participantes houver, mais segura é a cadeia de blocos(SEVEREIJNS, 2017).

Blockchains permissionadas públicas:

É uma blockchain distribuída, porém é controlada por participantes que possuem permissões restritas.

Estas são abordagens que contemplam contextos onde várias partes diferentes compartilham os mesmos dados e transações. Este tipo de *Blockchain* permite que haja um controle de acesso, ao mesmo tempo que possui uma grande visibilidade entre as partes permitidas (SEVEREIJNS, 2017).

3 Metodologia

3.1 Fluxo de trabalho

A principal metodologia escolhida para gerenciamento do projeto é o *KanBan*. Dentro do projeto, o *KanBan* é utilizado para a divisão das entregas principais e organização das tarefas para a conclusão de cada entrega. Através da organização em colunas, indicando o *status* de execução das partes do projeto, o *KanBan* provê uma grande flexibilidade no planejamento e diminui o número de gargalos criados pela centralização da responsabilidade de organização das entregas e atividades. Permite também a extração de métricas para mensuração do sucesso do projeto, através da priorização das entregas realizadas e pendências (RADIGAN, 2015).

A principal forma de utilização do *KanBan* é com um *board* de organização. A principal vantagem de sua utilização é a visualização fácil de todo o trabalho em progresso e, consequentemente, a fácil priorização de atividades e acompanhamento do fluxo de execução do projeto. Esta vantagem também gera um dos principais desafios de utilização desta metodologia, que é a necessidade de gerência e atualização contínua do trabalho em progresso (AHMAD JOUNI MARKKULA, 2013). Em outras palavras, é preciso que a responsabilidade pela manutenção do *board* do *KanBan* esteja clara para todos os membros da equipe que estão utilizando a metodologia.

Assim sendo, essa metodologia foi adotada por possuir um alto poder de organização e visualização da situação do projeto. Além disso, por sua simplicidade a utilização do *KanBan* se torna interessante no contexto de desenvolvimento de um Trabalho de Conclusão de Curso, onde geralmente o gerenciamento e desenvolvimento do projeto fica à cargo de um único responsável.

3.1.1 Ferramentas para gerenciamento de atividades

Zenhub: É um plugin para o Github utilizado para a criação de um *KanBan* online associado a um determinado repositório. Foi a principal ferramenta de gerenciamento. É colaborativa e fácil de usar, e permite que todos os participantes tenham uma visão unificada da situação do projeto.

Github: É uma ferramenta para gerenciamento de repositórios e versionamento de código, é utilizada para controle de versão durante o desenvolvimento deste trabalho e da solução. É usada também para acesso ao *board* de atividades no *Zenhub*.

3.1.2 Abordagem de desenvolvimento

Para a realização da etapa de desenvolvimento deste trabalho será utilizada uma abordagem baseada nos seguintes passos:

1. Definir um conjunto de requisitos que caracterizem uma infraestrutura segura para votações;
2. Elencar aqueles que serão implementados na solução proposta, e que são possíveis de serem realizados entro do escopo do projeto;
3. Definir os principais aspectos técnicos da construção da solução, tais como as tecnologias a serem utilizadas na construção desta;
4. Validar se os requisitos selecionados para serem implementados realmente são possíveis de serem concluídos;
5. Justificar os requisitos que não serão implementados na solução. Explicando o fator de complexidade ou escopo que não permite que estes sejam cumpridos neste trabalho;
6. Modelar uma solução de *software* que implemente os requisitos definidos no **passo 2**;
7. Utilizando a metodologia *KanBan*, limitar o escopo e dimensionar tarefas a serem realizadas baseado na prioridade dos componentes principais da solução;
8. Implementar as partes da solução de acordo com as tarefas definidas, e utilizar o *KanBan* para acompanhamento direto do progresso de desenvolvimento e priorização das atividades;

3.2 Requisitos gerais

De acordo com os principais aspectos definidos pelos desenvolvedores e pesquisadores de sistemas de votação eletrônicos, é possível definir um conjunto principal de requisitos que caracterizam um sistema de *E-voting*. A aplicação dos requisitos é uma tarefa complexa, e se totalmente implementados, constituiriam uma infraestrutura completa e segura para realização de votações em diversos âmbitos. Esses requisitos são:

- **Requisito 01:** Garantir que cada votante só possa realizar um voto;
- **Requisito 02:** Somente usuários registrados e devidamente autenticados podem votar;

- **Requisito 03:** Os resultados adquiridos ao longo da votação não devem ser publicamente divulgados antes do fim da votação;
- **Requisito 04:** Os votos precisam ser auditáveis;
- **Requisito 05:** Não deve ser possível associar um voto ao usuário que o realizou;
- **Requisito 06:** Os servidores precisam ser auditáveis;
- **Requisito 07:** Deve haver uma forma de autenticação segura do usuário, de modo que não seja possível associar um voto às informações de autenticação recebidas;
- **Requisito 08:** Deve haver o mínimo de outras partes envolvidas no processo de votação;
- **Requisito 09:** Deve-se manter os registros de votos por um período de tempo suficientemente grande para que sejam realizados processos de auditoria, caso necessário. O período de armazenamento dos registros deve ser pequeno o suficiente para evitar o vazamento de informações confidenciais;
- **Requisito 10:** Deve-se assegurar a segurança física da infraestrutura;
- **Requisito 11:** Deve-se assegurar a confiabilidade do hardware sendo utilizado;
- **Requisito 12:** O código da solução e todas as especificações e protocolos utilizados devem ser abertos, e facilmente auditáveis por qualquer parte interessada;
- **Requisito 13:** Deve possuir uma alta escalabilidade e uma alta disponibilidade, uma vez que o número de usuários pode ser muito grande.
- **Requisito 14:** O usuário deve conseguir auditar seu voto, e garantir que este foi devidamente contado;
- **Requisito 15:** O usuário deve conseguir verificar seu voto, e garantir que este não foi alterado;
- **Requisito 16:** Deve-se assegurar que seja possível descobrir quando algum dos dados foi alterado;
- **Requisito 17:** O sistema deve conseguir se recuperar de possíveis fraudes e ataques, de modo que não seja necessário realizar novamente todo o processo de votação;
- **Requisito 18:** O processo de votação deve ser flexível e permitir que os usuário possam votar através de vários tipos de dispositivos.

3.3 Solução proposta

3.3.1 Resultados esperados

Baseando-se nos requisitos gerais para construção de uma infraestrutura completa de *e-voting*, os requisitos que a solução proposta busca atender são:

- **Requisitos a serem cumpridos pela solução proposta**
  - ☑ É preciso garantir que cada votante só possa realizar um voto;
  - ☑ Somente usuários registrados e devidamente autenticados podem votar;
  - ☐ Os resultados adquiridos ao longo da votação não devem ser publicamente divulgados antes do fim da votação;
  - ☑ Os votos precisam ser auditáveis;
  - ☑ Não deve ser possível associar um voto ao usuário que o realizou;
  - ☐ Os servidores precisam ser auditáveis;
  - ☑ Deve haver uma forma de autenticação segura do usuário, de modo que não seja possível associar um voto às informações de autenticação recebidas;
  - ☐ Deve haver o mínimo de outras partes envolvidas no processo de votação;
  - ☐ Deve-se manter os registros de votos por um período de tempo suficientemente grande para que sejam realizados processos de auditoria, caso necessário;
  - ☐ Deve-se assegurar a segurança física da infraestrutura;
  - ☐ Deve-se assegurar a confiabilidade do hardware sendo utilizado;
  - ☐ O código da solução e todas as especificações e protocolos utilizados devem ser abertos, e facilmente auditáveis por qualquer parte interessada;
  - ☑ Deve possuir uma alta escalabilidade e uma alta disponibilidade, uma vez que o número de usuários pode ser muito grande.
  - ☑ O usuário deve conseguir auditar seu voto, e garantir que este foi devidamente contado;
  - ☑ O usuário deve conseguir verificar seu voto, e garantir que este não foi alterado;
  - ☐ Deve-se assegurar que seja possível descobrir quando algum dos dados foi alterado;
  - ☑ O sistema deve conseguir se recuperar de possíveis fraudes e ataques, de modo que não seja necessário realizar novamente todo o processo de votação;
  - ☐ O processo de votação deve ser flexível e permitir que os usuário possam votar através de vários tipos de dispositivos.

3.3.2 Ferramentas para desenvolvimento da solução

Ethereum:

Como discutido na subseção 2.1.1.2, para assegurar que o mínimo de outras partes estejam envolvidas é importante garantir que o sistema de armazenamento dos votos não estejam sobre o controle centralizado de uma única entidade, ou grupo de envolvidos. Para tal, uma boa abordagem é a utilização de uma rede de BlockChain não permissionada.

Com a utilização de uma rede não permissionada, existe a premissa de que qualquer um pode se tornar um nó participante da rede e armazenar os votos de forma distribuída. Isso potencialmente aumenta a confiabilidade dos dados.

As duas aplicações mais utilizadas, baseadas em *Blockchains* não permissionadas, são as criptomoedas *Bitcoin* e *Êthereum*.

Como o contexto de aplicação desta solução está relacionado à realização de votações, é preciso adaptar a utilização das transações e do consenso utilizado na *Blockchain*. A rede *Bitcoin* implementa uma versão simplificada de *smart contracts*, que permite a utilização de *scripts* programáveis para determinar como as transações devem se comportar.

Smart Contracts, ou *contratos inteligentes*, são programas seguros que representam acordos executáveis dentro da rede *Blockchain*. Estes contratos são utilizados por aplicações distribuídas baseadas em *Blockchain* para garantir acordos entre os nós com necessidade mínima de confiança entre os nós e dependabilidade de terceiros (BRAGA; SANTOS, 2017).

A linguagem de *scripting* da *Bitcoin* possui algumas limitações decorrentes de sua implementação. A primeira é que esta não suporta a utilização de estruturas de repetição. Esta característica possui o objetivo de evitar que ocorram laços infinitos durante o processo de verificação de transações. Consequentemente, isso pode fazer com que determinadas soluções sejam ineficientes em relação ao espaço utilizado (Ethereum White Paper, 2014).

Além disso dentro da rede da *Bitcoin* as transações só podem estar em dois estados específicos, sendo 'gastas' ou 'não gastas'. Isso torna difícil a implementação de aplicações baseadas em múltiplos estados.

Por último, as transações não conseguem ter acesso aos dados da *Blockchain*. O que limita diversas aplicações, uma vez que limita os dados sendo enviados de informações valiosas armazenadas nos blocos.

A Plataforma Ethereum, por outro lado, possui internamente uma linguagem de programação completa, que permite que os desenvolvedores possam programar contratos para realizar praticamente qualquer lógica que pode ser definida matematicamente. Isso

permite que a rede *Ethereum* possa ser utilizada em uma gama grande de aplicações. Desde a construção e utilização de tipos específicos de transações, até o desenvolvimento de outras criptomoedas que funcionam como contratos *Ethereum*.

Dentro da rede *Ethereum*, as características da *Blockchain* não interferem no funcionamento dos contratos. Isso implica que toda a lógica de restrição e correto funcionamento dos contratos deve estar definida dentro do próprio contrato. Isso caracteriza que o contrato não pode ser censurado ou restringido, e que o funcionamento deste dentro da rede será garantido uma vez que este esteja corretamente definido, e enquanto hajam taxas computacionais sendo pagas.

A plataforma *Ethereum* possui nativamente suporte integrado ao uso de *smart contracts*. Dentro da rede *Ethereum*, os nós formam uma estrutura computacional distribuídai, denominada *Ethereum Virtual Machine*, que exeutam os contratos (BRAGA; SANTOS, 2017). Por tais característica a plataforma Ethereum possui uma grande flexibilidade, e permite que se utilizem os contratos como base para construção de aplicações.

4 Resultados

4.1 Solução implementada

4.1.1 Tecnologias utilizadas

As principais ferramentas de *software* utilizadas no desenvolvimento da solução foram:

- Solc(*Solidity compiler*): Compilador da linguagem *Solidity*, linguagem na qual são escritos os contratos utilizados nas redes *Ethereum*;
- Web3py: Biblioteca python desenvolvida para interações com a rede Ethereum através do uso de protocolos baseados em *Remote Procedure Calls*. Foi utilizada para criar as transações na rede, bem como para compilar e submeter os contratos *Ethereum*;
- Python Flask: A biblioteca *Python Flask* foi utilizada para definição de uma API Rest simples, que disponibiliza *endpoints* utilizados para a criação das transações na *Blockchain*;
- Docker e Docker-Compose: A aplicação utilizada para compilar e submeter os contratos e a API para interação com a rede foram modelados como serviços baseados em containers *Docker*. Foi utilizado o docker-compose para orquestração dos serviços;
- Geth: É uma biblioteca desenvolvida na linguagem GO que implementa os protocolos da rede *Ethereum*. Foi utilizada para criação das contas, instanciação dos nós e configuração da *BlockChain* privada;

4.1.2 Arquitetura

A arquitetura definida é de um conjunto de microserviços que, juntos, formam um *framework* de votação. O objetivo é que este *framework* de votação seja utilizado por aplicações que já possuem um processo de votação definido e desejam utilizar a rede *BlockChain* para adicionar propriedades de auditabilidade e verificabilidade ao processo já existente.

Foram tomadas algumas decisões em relação à modelagem e objetivo da solução implementada. A primeira delas foi em relação à utilização de uma *Blockchain* híbrida em

vez de uma das *Blockchains* públicas padrões ou de uma das redes de teste. Esta decisão foi tomada a fim de obter algumas vantagens:

1. Ter um maior nível de controle sobre as permissões da rede;
2. O custo de criação das transações na rede pode ser mitigado, uma vez que a quantidade de *Ether* necessária para a realização do processo de votação pode ser alocada previamente para as contas que criarão as transações de votos;
3. As transações sendo processadas na rede serão apenas as relacionadas ao contrato de votação, e desta forma, os nós não precisarão gastar recursos de processamento com transações que não fazem parte do contexto em questão;

A rede híbrida foi implementada utilizando uma arquitetura de microserviços baseados em *Docker*. Esta arquitetura está representada na figura 5 e é composta de 4 tipos de serviços: *Miner Nodes*, *Boot Nodes*, *Compiler* e *Rest API*.

4.1.2.1 Miner Nodes

São os nós principais dentro da rede, agirão como nós completos, sendo responsáveis pela sincronização de todos os dados e transações da rede.

Entre os *Miner Nodes* também estão os *Sealers*, os nós responsáveis pela validação das transações e criação dos blocos na rede se alternando em períodos. Através dos *Sealers*, serão disparadas as interações de ativação do contrato na *BlockChain*.

Através dos *Miners Nodes* também são disponilizadas as API's RPC, que disponibilizam os *endpoints* para conexão utilizando os *clients Ethereum*. Estes endpoints permitem que qualquer um possa se conectar à rede e sincronize todas as transações e votos, tornando o processo de auditabilidade mais democrático e seguro.

4.1.2.2 Boot Nodes

É possível se conectar à uma determinada rede *BlockChain* usando diversas estratégias e uma delas é através do endereço 'enode' gerado por cada nó que já está dentro da rede. Desta forma, um nó que deseje participar da criação de blocos ou sincronização dos dados pode se conectar à partir de qualquer um dos nós pré-existentes na rede.

Outra forma é descrever nós estáticos que serão sempre utilizados para conexão na *BlockChain*. Estes nós possuirão um endereço fixo que será utilizado pelos nós que se conectarem posteriormente (Felix Lange, 2017).

A terceira abordagem que foi utilizada na solução desenvolvida neste trabalho foi a de utilizar um *BootNode*, um nó de referência que serve para facilitar a descoberta dos

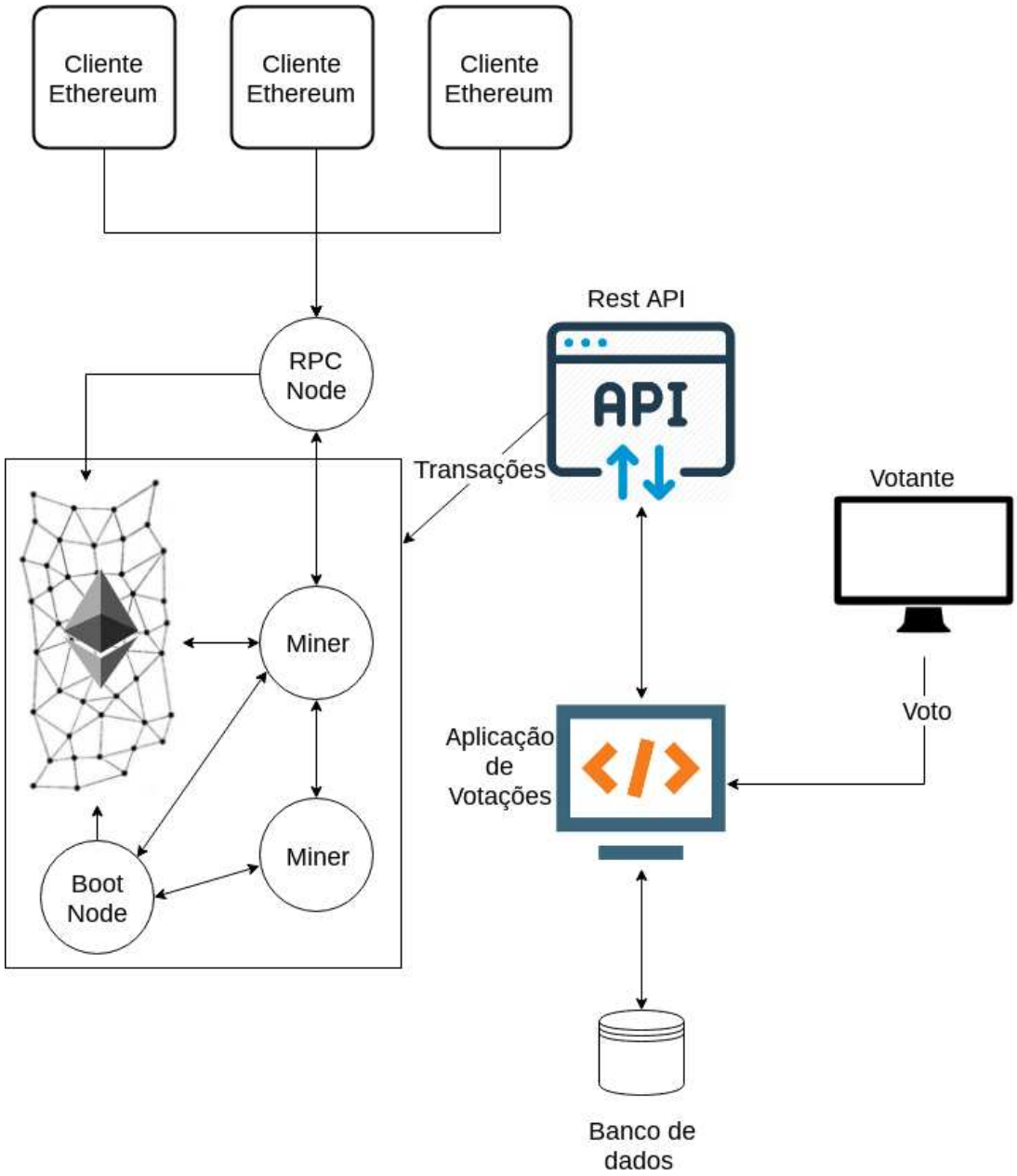

Figura 5 – Arquitetura do framework de votação

nós uns pelos outros. Caso os nós estejam dentro da mesma rede e utilizando as mesmas configurações, estes serão capazes de descobrir uns aos outros automaticamente, porém pode-se utilizar um *BootNode* como referência para garantir que os nós se encontrarão rapidamente durante a inicialização da *BlockChain*. Isso garantirá uma maior consistência desde os primeiros votos que possam ser realizados na rede.

Os *Miner Nodes* e os nós completos se conectam através do endereço de *BootNode* definido e este compartilha os metadados entre os nós conectados na rede, facilitando o início do processo de criação de blocos.

4.1.2.3 Compiler

Serviço utilizado para compilar e submeter o contrato para dentro da rede *Ethereum*.

A partir do código do contrato, este serviço utiliza uma das contas base definidas para compilar o contrato, criar um transação de *deploy* e, em seguida, instanciar uma interface do contrato que será utilizada para interagir com as funções do contrato dentro da *BlockChain*.

4.1.2.4 Rest API

Este serviço é a interface de comunicação com a *BlockChain* e é implementado como uma API *Flask* com 4 *Endpoints* principais:

- cast_vote - POST: *Endpoint* utilizado para adicionar um voto a um determinado candidato. Recebe como parâmetro da requisição o nome do candidato em que se quer votar;
- add_candidate - POST: *Endpoint* utilizado para adicionar um candidato. Recebe como parâmetro o nome do candidato que será adicionado como uma opção de votação;
- add_voter - POST: *Endpoint* para dar permissão de voto à uma determinada conta. Recebe como parâmetro o endereço da conta na rede *Ethereum* que possuirá permissão para depositar um voto;
- show_election_results - GET: *Endpoint* utilizado para mostrar o estado atual do processo de votação, e os votos que cada candidato possui até então;

4.1.3 Workflow de funcionamento

Supondo que exista uma plataforma chamada **VoteX** que realiza processos de votação para eleição de representantes locais. Nesse caso, o fluxo de submissão de um

voto se daria com as seguintes etapas:

1. O *framework* de votação seria inicializado e configurado com as contas base que possuem permissão de criar transações definidas dentro do contrato. O contrato de votação também seria compilado e enviado para a *BlockChain*. Depois disso, teríamos uma rede *Ethereum* local funcionando com alguns nós mineradores e uma API Rest de votação que será utilizada pelo **VoteX** para interagir com o contrato criado;
2. Durante a criação do processo de votação no **VoteX**, todos os usuários que possuem permissão para participar da votação seriam inseridos no contrato com permissão de votantes. Em seguida, todos os candidatos válidos seriam inseridos no contrato como candidatos válidos;
3. O usuário se logaria na plataforma **VoteX** e iria à página de votação da própria aplicação, escolheria votar no candidato Z e submeteria sua opção de voto;
4. O **VoteX** faria uma requisição para a API de votação que, por sua vez, criaria uma transação na rede solicitando ao contrato que fosse adicionado um voto ao candidato Z;
5. O contrato conferiria se o usuário tentando realizar um voto possui permissão para tal e se este usuário ainda não votou. Após a conferência, seria verificado se o candidato a que se destina o voto é um candidato válido. Caso estes requisitos sejam cumpridos, o número de votos que um candidato possui é incrementado e a permissão do votante é retirada pois este já utilizou seu voto.

4.1.4 Configuração da rede

Toda rede *BlockChain* é iniciada com um bloco chamado **Genesis Block**. Quando a *BlockChain* é executada com as configurações padrão, o bloco Genesis principal da rede em questão é sincronizado e adiciondo à cadeia de blocos. No contexto de redes privadas, geralmente o bloco inicial possui configurações diferentes que se adequam às características da rede.

A configuração da rede é feita através de um arquivo chamado **genesis.json** que define o Genesis Block que, por sua vez, define as características da rede.

As informações definidas no Genesis Block da solução foram os seguintes:

- **chainId**: Este fator define um número identificador da rede. Os nós que irão se conectar a esta rede devem fornecer esse parâmetro no momento da conexão. Isso ajuda a restringir o acesso à rede somente aos nós que possuem tal idenficador;
- **homesteadBlock**: Indica a *release* do *Ethereum* sendo utilizada;

- **clique**: Este parâmetro indica que a rede está utilizando um protocolo de consenso baseado em Prova de Autoridade. É aqui onde é definido o período de criação dos blocos, que indica o tempo mínimo que deve haver entre o *timestamp* de dois blocos consecutivos;

- **extraData**: Campo onde estão definidos os endereços dos *Sealers* autorizados a criar blocos, participar no consenso e validar transações;

- **gasLimit**: É o valor máximo que pode ser gasto em um bloco. Cada transação possui um número de *gasLimit* correspondente e a soma do valor limite de cada transação não deve ser superior ao valor total do bloco. Este parâmetro é utilizado para impedir que um número gigante de transações seja validado em um único bloco, o que daria margem para ataques de nós maliciosos tentando alterar a condição de um número grande de transações por bloco;

- **alloc**: Neste atributo pode-se inicializar uma determinada quantidade de *Ether* no conjunto de contas definidas. No contexto deste trabalho, foi utilizada para limitar as contas que serão capazes de criar transações e ativar o contrato de votação.

4.1.5 Permissionamento na rede

4.1.5.1 Criação de blocos

A criação de blocos e o processamento das transações se dá através da utilização de um protocolo de consenso conhecido como Prova de Autoridade. Nesse processo, os nós que possuem permissão de mineração são conhecidos como *Sealers* e se alternam no processo de validação das transações e criação dos blocos. A cada período de tempo, um nó que possui permissão valida um conjunto de transações e propõe a adição de um novo bloco na cadeia.

Um dos fatores que traz sentido à utilização deste mecanismo em contextos de *BlockChains* privadas ou mistas é que não é preciso que os nós resolvam problemas matemáticos complexos como prova de confiabilidade. No contexto de redes baseadas em prova de trabalho, há um custo computacional significativamente menor já que a confibilidade vem da restrição que apenas um número limitado de nós possui permissão na criação de blocos(Karalabe, 2017).

Há duas formas principais de se garantir acesso à um nó na rede: a primeira delas é definir no bloco inicial da rede quais endereços estão autorizados a minerar transações e a segunda forma é que mais da metade dos nós proponham a adição de um novo nó com permissão à rede.

A primeira alternativa consiste em definir estaticamente a assinatura dos nós que podem agir como *Sealers* criando blocos na rede.

Para a segunda abordagem, basta que mais da metade dos participantes da rede proponham que um novo nó seja aceito no consenso para que este ganhe permissão de mineração. Isto faz com que os nós que possuem permissão queiram manter uma boa reputação na participação.

Além disso, a utilização do mecanismo de Prova de Autoridade permite que os participantes possam retirar a permissão de *Sealer* de um determinado nó, caso este esteja agindo de maneira maliciosa na rede. Para tanto, basta que mais da metade dos nós da rede requisitem a expulsão de um nó do consenso(Karalabe, 2017).

Dentro de uma rede executando um consenso baseado em Prova de Autoridade é possível observar a seguinte sucessão de atividades:

1. São definidos os nós que podem participar do consenso de validação dos blocos e o período de criação destes;
2. A cada período de tempo, um dos nós é escolhido via *Round-Robin* e recebe a permissão para validar um conjunto de transações. Em seguida, tenta criar um novo bloco;
3. O nó que possui a prioridade para criação no turno é o preferido para criação do bloco. Caso este nó não proponha um bloco, outros nós podem tentar propôr blocos, porém estes terão menor prioridade em relação ao nó da vez;
4. Depois que um nó com permissão assina e propõe um determinado bloco, ele não pode assinar os próximos n/2 blocos. Isto serve para que um nó malicioso não tente sobreescreever os votos de nós honestos atuando para removê-lo da rede.

4.1.5.2 Criação de transações

Como é preciso pagar taxas para execução das transações, caso fosse utilizada alguma das redes de teste oficiais ou uma das redes *Ethereum*, seria necessário que as contas tivessem uma certa quantidade de *Ether* para pagar pelas taxas.

Como o objetivo é incentivar que o maior número de pessoas participe do consenso sem previsão de recompensas monetárias reais, se torna necessário eliminar o custo de criação das transações. Além disso, esta solução visa implementar uma arquitetura que possa ser utilizada em contextos menores, ou em processos de votação mais baratos onde não há recursos para serem gastos com o pagamento da execução de contratos *Ethereum*.

A estratégia aqui utilizada foi utilizar um número pré-definido de contas base que serão utilizadas para criar as transações, de formar que todas as transações sejam feitas à partir destas contas. É possível utilizar uma diretiva chamada **alloc** dentro do Genesis Block para pré-alocar uma determinada quantidade de *Ether* para as contas com os endereços definidos. E.g.:

```
"alloc": {
        "7df9a875a174b3bc565e6424a0050ebc1b2d1d82":
                { "balance": "300000" },
        "f41c74c9ae680c1aa78f42e5647a62f353b7bdde":
                { "balance": "400000" }
}
```

Quando um voto vai ser depositado na *BlockChain*, valida-se previamente a permissão do usuário tentando criar a transação: caso este usuário possua permissão de votante, será então criada um requisição de voto à API, e em seguida a seleção de uma das contas base e a partir da qual será criada a transação.

Esta estratégia está relacionada à resolução do **Requisito 05**, uma vez que, a partir da utilização de um conjunto pré-definido de contas, as transações não estarão associadas diretamente à conta de nenhum dos usuários participando do processo de votação.

4.1.5.3 Autenticação e permissionamento de usuários

Há três tipos de transações principais que podem ser criadas nesta rede e que demandam cuidado em relação à permissão de criação de transações:

1. **Adição de candidatos**: É preciso que haja um cuidado com relação aos candidatos adicionados à rede, uma vez que caso um candidato exista no mapa de candidatos do contrato, um usuário poderia criar uma transação adicionando votos à este candidato.

   A estratégia utilizada é adicionar, durante a instanciação do contrato, um conjunto de contas pré-autorizadas que terão permissão para adicionar candidatos ao contrato. Levando em consideração a propriedade de imutabilidade dos contratos na *BlockChain*, uma vez que estas contas são definidas no começo da execução do contrato, somente essas contas poderão alterar o estado do mapa de candidatos adicionando novas opções ao processo de votação.

2. **Adição de votantes**: Para que haja um processo de votação justo, é preciso que exista uma forma de garantir que apenas usuários autorizados poderão criar transações de voto na rede. Esta necessidade está relacionada ao **Requisito 02** definido anteriormente.

   Caso o contexto sendo implementado utilizasse uma *BlockChain* pública, criar uma estratégia de autenticação para restringir quem pode ou não interagir com o contrato seria uma tarefa extremamente complexa, uma vez que não é possível limitar a criação de transações na rede *Ethereum* e, qualquer um que possua a quantidade

de *Ether* suficiente para criar a transação, poderia executar o contrato e criar um voto.

O contexto aqui desenvolvido é um contexto híbrido, onde existe um certo nível de privacidade em relação às permissões na rede, porém a solução ainda é baseada no princípio de que o acesso aos dados da rede deve ser público para garantir um processo de votação mais democrático e seguro, uma vez que qualquer um poderá sincronizar os dados da rede e auxiliar no processo de auditoria dos dados.

Nesta solução foi adotado um modelo de autenticação por terceiros, onde a aplicação que estiver utilizando a *BlockChain* no seu processo de votação estará responsável por garantir que o usuário votante é real e possui permissão de voto em um determinado processo. Esta forma de autenticação é mais simples, uma vez que caso a aplicação utilizando a *BlockChain* já possua a funcionalidade de gerenciamento de usuários, só será preciso que os votantes recebam permissão de voto dentro do contrato.

Através do *endpoint* para adicionar um votante, é possível ativar no contrato uma função dar permissão à um usuário a partir de uma chave 'única', criada com uma *hash* gerada a partir do nome, da senha cifrada e do id do usuário na aplicação.

```
function addVoter (string _voterKey) public {
        require(allowed_accounts[msg.sender]);

        voters[keccak256(abi.encodePacked(_voterKey))] = true;
}
```

Na estratégia proposta neste trabalho, a autenticação dos usuários e o permissionamento na rede são feitos pela aplicação que está utilizando as API's e interagindo com a *BlockChain*, sendo assim o permissionamento dos usuários dependerá do contexto onde está sendo utilizada a infraestrutura de votação. Deve ser criada uma política de forma que somente um número seleto de usuários mais privilegiados possa criar requisições.

Assume-se que a aplicação utilizando a rede estará agindo de maneira honesta, uma vez que esta é a principal interessada em assegurar o processo de votação. Dessa maneira, não há perdas significativas em ter essa terceira parte participando do processo de segurança da abordagem de votação.

Esta abordagem não traria grande impacto para o usuário final, uma vez que este continuaria utilizando a aplicação de voto normalmente, sem que sua usabilidade fosse diretamente afetada.

3. **Adição de votos**: O **Requisito 01** deste projeto demanda a garantia de que cada usuário só pode votar uma vez em um mesmo processo de votação.

A função abaixo é executada durante o processo de submissão de voto.

```
function vote (string _candidateName, string _voterKey) public {
    // Exige que o votante tenha permissão para votar
    require(voters[keccak256(abi.encodePacked(_voterKey))]);

    // Exige que o votante não tenha votado anteriormente
    require(!has_voted[keccak256(abi.encodePacked(_voterKey))]);

    uint _candidateID = candidates_ids[_candidateName];
    // Verifica que o candidato é um candidado válido
    require(candidates[_candidateID].definedCandidate == true);

    // Atualiza o número de votos do candidato
    candidates[_candidateID].voteCount ++;

                // Define que o votante em questão já votou
    has_voted[keccak256(abi.encodePacked(_voterKey))] = true;
}
```

Esta função recebe como parâmetro o nome do candidato e a *hash* gerada a partir dos dados do usuário. Será então verificado se o usuário é um votante dentro do contrato, se este votante ainda não utilizou seu voto e se o candidato a que se destina o voto é um candidato válido dentro do contrato. Caso todas as condições anteriores sejam verdadeiras, o número de votos recebidos pelo candidato será incrementado e o mapa de votantes será atualizado, indicando que o votante em questão já utilizou o seu voto e, portanto, não possui mais permissão para votar. Caso alguma das condições falhe, o contrato disparará um erro de permissão de execução da função.

4.1.6 Consistência de dados

Na rede *Ethereum*, existe uma unidade de medida chamada 'GAS' que determina o custo do esforço computacional. De acordo com o tipo de transação ou contrato sendo executado, o custo em unidades de 'GAS' é diferente. Esta unidade de medida existe para que o custo de se realizar computações na rede não esteja diretamente relacionado ao valor atual da moeda *Ether*, pois o custo computacional não oscila na mesma proporção.

Poderá haver um contrato específico que apenas pessoas com devidas permissões podem executar. Este será utilizado para prover uma determinada quantidade de *Ether* à conta de um votante. Esta determinada quantidade de Ether representará a capacidade de votar uma vez.

Cada conta *Ethereum* possui um valor global chamado *nonce* que é acessível à todos os nós. Este valor é sempre verificado pelos nós durante a validação de uma transação, e apenas a transação que corresponde ao valor de *nonce* atual será processada. Desta forma, o *Ethereum* garante que uma mesma quantidade de *Ether* não pode ser gasta mais de uma vez.

Dentro da rede *Ethereum*, a validação de uma transação segue os passos abaixo (Ethereum White Paper, 2014):

- É conferido se a transação possui a formatação correta, se a assinatura da transação é válida, e se o valor de *nonce* da transação é o mesmo valor identificado na conta do usuário;
- A taxa da transação é calculada como STARTGAS * GASPRICE. A taxa é subtraída do saldo da conta do remetente e o valor de *nonce* da conta do remetente é incrementado. Caso o saldo não seja suficiente, a transação é invalidada;
- É inicializado o GAS com o valor de STARTGAS, e é retirada uma quantidade de GAS correspondente ao preço por byte a ser pago na transação;
- Caso a conta de destino seja uma conta externa, o valor da transação é enviado à conta de destino. Caso a conta de destino seja um contrato, este é executado até ser completo ou até o GAS da transação se esgotar;
- Caso a transferência tenha falhado, porque o remetente não possui dinheiro suficiente ou porque a transação esgotou todo o seu GAS, então todas as operações realizadas são revertidas, exceto pelo pagamento das taxas de criação da transação e das taxas do nó minerador;
- Caso contrário, o GAS excedente é devolvido para o remetente, e as taxas pelo GAS consumido são enviados ao nó minerador.

O valor de *nonce* evita ataques de repetição em que o atacante tente propositalmente gastar várias moedas ou alterar a ordem das transações aumentando a recompensa por uma determinada transação.

Desta forma, com as propriedades de validação de transações da rede *Ethereum*, e com a utilização de um contrato que garanta que cada conta só possa receber o valor correspondente à um voto, é possível garantir que um votante não pode votar mais de uma vez. Isso valida o cumprimento do **Requisito 01**.

4.1.7 Auditabilidade

Na infraestrutura proposta neste trabalho, ainda serão explorados os benefícios de autabilidade de uma rede pública. Apesar de nem todos os nós participantes da rede poderem criar blocos e validarem transações, qualquer nó poderá ser juntar à rede e sincronizar os dados com o histórico das transações de votações e criação de candidatos. Desta forma, qualquer interessado em auditar o processo de determinada aplicação precisará apenas utilizar algum cliente para se conectar à rede e baixar os dados da *BlockChain*.

Através do nó principal, será disponibilizado um *endpoint* que atua como um ponto de conexão para nós que desejam começar a partipar do consenso na rede. A partir desse *endpoint*, qualquer *client Ethereum* pode ser utilizado para que o nó entre no consenso na rede e comece a sincronizar os dados da *BlockChain* ou a criar blocos de transações, caso esse nó possua a pemissão para isso.

Como foi definido que nessa infraestrutura o voto dos usuários serão representados como transações, é inerente a capacidade que os votantes terão de verificar que suas transações foram devidamente realizadas, e que os votos foram devidamente destinados aos candidatos corretos. Isto torna possível a auditabilidade dos votos e a verificação da integridade destes.

Os **Requisitos 04 e 15** referem-se à auditabilidade dos votos realizados. A implementação destes requisitos é uma característica natural da implementação das *Blockchains* públicas. No caso do *Ethereum*, existem diversas ferramentas para visualização do *status* das transações na *ledger*, o que garante a sua devida realização.

Como a infraestrutura proposta neste trabalho é construída em cima de uma *ledger* privada com acesso público, o processo de fraude de dados se torna extremamente trabalhoso. O primeiro fator que impõe impedimentos a fraudes é o acesso restrito à criação de transações, que é permitida somente aos nós que possuem contas permitidas. O segundo aspecto é que pode haver um grande número de nós mineradores participando do consenso.

Na solução implementada, quando uma transação de voto for criada com sucesso, será retornado a *hash* de identificação da transação. E.g.:

```
{"status":"success",
 "transaction_hash":
     "0xbaaf00a0d6d0a685d3f47949a03da4a125f3416cf"
}
```

Assim, o usuário poderá utilizar alguma ferramenta de gerencimento de contas *Ethereum* para para visualizar os detalhes e validar sua transação, garantindo que o seu voto foi realmente criado e destinado ao candidato correto.

Porém, ao mesmo tempo que a transparência das transações e da *ledger* possibilita a auditabilidade dos dados, esta também fragiliza em certos aspectos o sigilo do processo de votação, visto que não é possível garantir diretamente o sigilo dos resultados da votação.

Será possível o acompanhamento irrestrito das transações podendo destacar, por exemplo, qual candidato estaria recebendo mais votos. Isso poderia impactar diretamente na opinião dos outros votantes, ou servir de insumo para manipulações de opinião. Por este motivo, não se atendeu o **Requisito 03** na solução proposta.

4.1.8 Escalabilidade e disponibilidade

Uma das principais preocupações de redes *Blockchain* públicas é a escalabilidade da rede (Ethereum White Paper, 2014). O **Requisito 13** implica justamente que é de extrema importância garantir a escalabilidade e disponibilidade para contextos críticos como votações que envolvem um grande número de pessoas.

Por serem redes públicas, que permitem a qualquer pessoa se tornar um nó participante, as *Blockchains* públicas geralmente possuem como característica inerente uma grande disponibilidade. Porém, como consequência de seus protocolos de consenso, estas sofrem do efeito que o histórico das transações tem que ser armazenadas em todos os nós, o que faz com que o tamanho da **ledger** cresça em uma taxa alta. No caso da *Bitcoin*, esta taxa é, em média, de 1 *MegaByte* por hora. Caso haja um aumento repentino no número de transações sendo processadas, esta taxa pode se tornar ainda maior.

A rede Ethereum utiliza algumas soluções para tentar minimizar o problema do tamanho da cadeia de blocos. Uma delas é justamente a utilização da árvore de estados exemplificada na Figura 4, onde entre cada um dos blocos haverá apenas um pequena modificação de estados. Logo, todo o estado dos blocos ao longo do tempo pode ser representado através de uma estrutura de árvore chamada *Patricia Tree*. Assim, os nós que validarão os blocos podem verificar de forma eficiente cada estado, apenas tendo acesso à árvore que representa a cadeia de blocos em questão, sem a necessidade de baixar a cadeia inteira.

Como na implementação deste trabalho é utilizado o mecanismo de prova de autoridade, existe a preocupação em relação ao período de criação dos blocos, que deve ser ajustado de acordo com o número de participantes no consenso, já que, quanto menor o período, mais rápido os blocos serão criados e o tamanho da *ledger* crescerá em uma taxa mais acelerada.

4.1.9 Imutabilidade dos dados

A validação do **Requisito 17** está relacionada à forma como os blocos e transações são validados dentro da rede *Ethereum*. Este requisito refere-se à recuperação do estado

da *Blockchain* em caso de possíveis fraudes e ataques à rede.

Como mencionado, um aspecto interessante é que a segurança e "imutabilidade" das redes *Blockchain* não está na capacidade de garantir que os dados não podem ser efetivamente alterados, e sim reside na dificuldade que existe no processo de alteração destes dados.

Como descrito nos passos para validação de um bloco, a primeira etapa deste processo é verificar se o bloco anterior existe e se a *hash* de descrição do bloco corresponde ao valor de referência que existe no bloco atual. Logo, caso um nó malicioso deseje alterar o conteúdo de um bloco, este terá que recriar todos os blocos que foram criados após este bloco até o instante atual no tempo, pois esses blocos serão considerados inválidos pelos nós que estiverem validando a cadeia. Porém, no consenso por Prova de Autoridade, os nós se alternarão na criação dos blocos e, além disso, o nó criador do último bloco não pode assinar os próximos n/2 blocos, com n sendo o número de nós. Isto faz com exista uma grande dificuldade para o atacante assinar dois blocos consecutivos.

Levando-se em conta que a dificuldade da Prova de Autoridade cresce à medida que o número de nós validadores cresce, quanto mais a rede cresce, mais difícil se torna o processo de alterar um conjunto de blocos do instante X até o momento atual.

Como as transações são processadas, validadas e armazenadas por diversos nós na rede, e qualquer um pode tornar-se um nó participante, o **Requisito 06** não é possível de ser realizado diretamente pois não é possível auditar cada um dos nós individualmente. Ao mesmo tempo, os protocolos de consenso e o funcionamento da *Blockchain* garantem que os nós honestos criarão novos blocos de transações a partir da maior cadeia de blocos válida. E, como qualquer nó pode sincronizar localmente todo o histórico de blocos e auditar o seu conteúdo, é possível "validar" o conteúdo de todos os servidores, já que a cadeia de blocos sendo auditada é a mesma sendo utilizada pelos nós honestos. Consequentemente, preocupar-se com a segurança física da infraestrutura e a confiabilidade do hardware também se tornam fatores irrelevantes, uma vez que pode existir uma quantidade muito grande de nós processando as transações. Desde que a maioria dos nós processando as informações sejam nós honestos, os protocolos de consenso funcionarão bem, independemente das condições anteriores.

4.1.10 Disponibilidade dos dados

No que diz respeito à disponibilidade dos dados, como o paradigma de funcionamento da *Blockchain* é justamente baseado em registros ao longo do tempo, as informações não serão apagadas da *ledger* e estarão sempre disponíveis para auditabilidade, desde que haja algum nó agindo como *full node*, que tenha sincronizado todos os dados da *BlockChain*.

4.1.11 Acesso à implementação

A solução foi construída utilizando como base a plataforma *Ethereum*, que é *Open Source*, bem como todo o código da solução que utiliza a licença GPL 3 (GNU, 2007), e portanto também um *software* livre.

Porém, devido à complexidade da implementação do client *Ethereum* e da própria *ledger*, não é possível garantir que qualquer pessoa possa verificar a implementação e garantir sua segurança. Apenas pessoas com perfil e conhecimento técnico terão capacidade para avaliar com profundidade o código das ferramentas utilizadas na construção da solução proposta.

Todo o código deste trabalho pode ser encontrando no seguinte repositódio do *GitHub*: <https://github.com/MatheusMiranda/ethereum-voting>.

5 Conclusões

O objetivo deste trabalho é validar a utilização de uma rede *BlockChain* em um contexto de votação e implementar uma infraestrutura para votações onde se pudesse garantir alguns requisitos de segurança, principalmente em relação à integridade e auditabilidade dos votos.

Foi modelado e desenvolvido um projeto que pode ser acoplado em contextos de aplicações que já possuam um processo de votação, com o foco em ser extensível e escalável de acordo com a necessidade.

O projeto aqui desenvolvido foi implementado com uma rede híbrida, com o objetivo de criar um determinado nível de permissionamento e ter mais liberdade nos protocolos de consenso e validação utilizados. Utilizar uma das redes públicas oficiais também implicaria na necessidade de gastos reais com a compra de *Ether* para execução das transações e *deploy* do contrato.

A utilização de contas base pré-configuradas com determinada quantidade de *Ether* na inicialização da *BlockChain* garante que todas as transações da rede são criadas de modo que não haverá dados do usuário relacionados ao voto. Porém, exige que haja um esforço inicial para criação e pré-configuração das contas base, o que demanda conhecimentos básicos da rede *Ethereum*.

Com a disponibilidade de um *endpoint* público para conexão de qualquer nó, é possível a livre conexão e sincronização dos dados da rede. Porém, apesar da possibilidade de auditoria, existe uma possibilidade muito baixa de que pessoas sem conhecimento técnico possam auditar os dados e as transações.

No geral, em relação a usabilidade, a solução implementada possui uma experiência ruim e exige um determinado nível de conhecimento prévio sobre redes *Ethereum*. Apesar disso, por estar organizada em um conjunto de micro serviços baseados em *Docker*, a solução é de fácil distribuição e reprodução. Além disso, pode ser utilizada em contextos de aplicações pequenas ou para testar contratos na rede *Ethereum* já que pode ser reproduzida em infraestruturas com poucos recursos.

Em relação à flexibilidade, caso se queira utilizar a mesma arquitetura em contextos diferentes, seria preciso alterar apenas o contrato sendo migrado na rede e os *endpoints* da API *Flask* para que ativassem as respectivas funções dentro do contrato em questão.

A disponibilidade dos serviços dependerá da infraestrutura onde está sendo executada a solução. Por exemplo, caso a quantidade de transações sendo requisitadas cresça muito, torna-se necessário implementar alguma estratégia para escalar o gerenciamento

de requisições por parte do serviço de API.

Já sobre a disponibilidade dos dados, quanto mais nós se conectarem à rede, maior será a disponibilidade das transações já que estes sincronizarão os blocos da rede. Porém, caso os serviços dos nós que fornecem o *endpoint* de conexão fique indisponível, não será possível que novos nós externos se conectem à rede e sincronizem os dados. Se os nós *Sealers*, responsáveis pela mineração dos blocos, ficarem indisponíveis a rede ficará estagnada e novas transações não poderão ser validadas já que não haverá protocolo de consenso sendo executado.

A infraestrutura implementada atende alguns dos requisitos propostos e pode ser utilizada em contextos onde a aplicação utilizando a rede está interessada na auditabilidade e verificabilidade do processo de votação. Alguns dos requisitos definidos foram implementados e testados em um contexto pequeno e o resultado da implementação deles foi um *framework* que pode ser extendido ou adaptado para utilização em outros contextos além de processo de votação.

As instruções para garantir uma boa escalabilidade e disponibilidade foram descritas neste trabalho, porém é necessário testar a solução desenvolvida em contextos mais complexos onde há uma escala maior de envolvidos. Concluiu-se, então, que implementar todos os requisitos é uma tarefa extremamente complexa e trabalhosa, portanto fora do escopo e dos recursos deste trabalho.

5.1 Trabalhos futuros

1. Não foi implementada uma política de utilização das contas base, definindo quais delas devem ser utilizadas em que momento. É preciso implementar um algoritmo *Round Robin* para utilizar as contas alternadamente de acordo com as requisições feitas;

2. A fim de criar um contexto com uma boa usabilidade para os votantes, é preciso implementar um cliente para facilitar a criação de contas e visualização das transações relacionadas à uma conta, que facilite a visualização dos estados do contrato de votação e como este é afetado pelas transações de voto criadas;

3. É necessário facilitar o permissionamento de nós mineradores, uma vez que, para adicionar novos nós após a inicialização da rede, é necessário que mais da metade dos *Sealers* solicitem o permissionamento para o novo nó;

4. Uma boa melhoria seria facilitar a criação de contas base depois da inicialização da *BlockChain*, criando uma forma de dar a estas contas uma determinada quantidade de *Ether* e permissão de criar transações;

5. Para tornar a utilização do framework ainda mais flexível seria proveitoso que existissem contratos com opções de votação mais flexíveis para serem utilizados em contextos mais comuns, como exemplo o de usuários fazendo votações de opinião em vários tópicos no mesmo processo.

Referências

AHMAD JOUNI MARKKULA, M. O. M. O. Kanban in software development: A systematic literature review. 2013. Disponível em: <http://ieeexpore.ieee.org/document/6619482/>. Citado na página 31.

ARROW, K. J. Social choice and individual values. 1951. Citado 2 vezes nas páginas 15 e 16.

BASHIR, I. *Mastering Blockchain*. [S.l.: s.n.], 2017. Citado 3 vezes nas páginas 26, 27 e 29.

Bitcoin Developer Guide. *Bitcoin Developer Guide*. 2014. Disponível em: < <https://bitcoin.org/en/developer-guide> >. Acessado em: 08 mai. 2018. Citado 5 vezes nas páginas 8, 22, 23, 28 e 29.

BOCHSLER, D. Can internet voting increase political participation? 2010. Disponível em: <https://www.eui.eu/Projects/EuDO-PublicOpinion/Documents/bochslere-voteeui2010.pdf>. Citado 3 vezes nas páginas 7, 15 e 16.

BRAGA, F. C. H. M. A. M.; SANTOS, R. R. dos. Segurança de aplicações blockchain além das criptomoedas. 2017. Disponível em: <https://sbseg2017.redes.unb.br/index.php/anais/>. Citado 11 vezes nas páginas 7, 19, 20, 21, 22, 23, 25, 26, 27, 35 e 36.

CACHIN, M. V. C. Blockchain consensus protocols in the wild. 2017. Disponível em: <https://arxiv.org/abs/1707.01873>. Citado na página 20.

Ethereum White Paper. *Ethereum White Paper*. 2014. Disponível em: < <https://github.com/ethereum/wiki/wiki/White-Paper> >. Acessado em: 15 mai. 2018. Citado 6 vezes nas páginas 7, 25, 26, 35, 47 e 49.

Felix Lange. *Geth static nodes*. 2017. Disponível em: < <https://github.com/ethereum/go-ethereum/wiki/Connecting-to-the-network> >. Acessado em: 15 jun. 2019. Citado na página 38.

GNU. *GNU GENERAL PUBLIC LICENSE 3*. 2007. Disponível em: < <https://www.gnu.org/licenses/gpl-3.0.pt-br.html> >. Acessado em: 18 out. 2018. Citado na página 51.

HANIFATUNNISA, B. R. R. Blockchain based e-voting recording system design. 2017. Disponível em: <https://ieeexplore.ieee.org/document/8272896/>. Citado na página 20.

HASSAN, X. Z. A. Design and build a secure e-voting infrastructure. 2013. Disponível em: <https://ieeexplore.ieee.org/document/6578240/>. Citado 3 vezes nas páginas 14, 17 e 18.

Karalabe. *Proof of Authority*. 2017. Disponível em: < <https://github.com/ethereum/EIPs/issues/225> >. Acessado em: 02 jul. 2019. Citado 2 vezes nas páginas 42 e 43.

KERRNEL. Internet voting: A requiem for the dream. 2016. Citado na página 14.

KERSTING, H. B. N. Electronic voting and democracy. 2004. Citado 3 vezes nas páginas 11, 13 e 14.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. 2008. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Citado 6 vezes nas páginas 7, 19, 23, 24, 25 e 29.

PAN, E. H. H.; ANSARI, N. Ensuring voters and candidates' confidentiality in e-voting systems. 2011. Disponível em: <https://ieeexplore.ieee.org/document/5876452/>. Citado na página 14.

Paul Cuff, Sanjeev Kulkarni, Mark Wang and John Sturm. *Voting Research - Voting Theory*. 2015. Disponível em: < <https://www.princeton.edu/~cuff/voting/theory.html> >. Acessado em: 25 mai. 2018. Citado na página 16.

RADIGAN, D. A brief introduction to kanban. 2015. Disponível em: <https://www.atlassian.com/agile/kanban>. Citado na página 31.

RAVAL, S. Descentralized application: Harnessing bitcoin's blockchain technology. 2016. Citado na página 11.

SEVEREIJNS, L. What is blockchain? how is it going to affect business? 2017. Citado 2 vezes nas páginas 29 e 30.

SPADA JONATHAN MELLON, T. P. P.; SJOBERG, F. Efects of the internet on participation - study of a public policy referendum in brazil. 2015. Disponível em: <https://www.changetomorrow.io/docs/prswp-digitalpb-peixoto-libre.pdf>. Citado na página 15.

SWAN, M. Blockchain: Blueprint for a new economy. 2015. Citado na página 11.

VUKOLIć, M. Rethinking permissioned blockchains. 2017. Citado na página 30.